

ENKCF-1.4

VAX 11/730 IDC
PART 1 MICRO

EP-ENKCF-DL-1.4
FICHE 1 OF 4

MAY 1983
COPYRIGHT © 82-83
MADE IN USA

00000000

ENKCF-1.4

VAX 11/730 IDC
PART 1 MICRO

EP-ENKCF-DL-1.4
FICHE 2 OF 4

MAY 1983
COPYRIGHT © 82-83
MADE IN USA



ENKCF-1.4

VAX 11/730 IDC
PART 1 MICRO

EP-ENKCF-DL-1.4
FICHE 3 OF 4

MAY 1983
COPYRIGHT © 82-83
MADE IN USA

discrete

ENKCF-1.4

VAX 11/730 IDC
PART 1 MICRO

EP-ENKCF-DL-1.4
FICHE 4 OF 4

MAY 1983
COPYRIGHT© 82-83
MADE IN USA

digital

IDENTIFICATION

Product code: ZZ-ENKCF VERSION 1.40

Product name: MICRO-DIAGNOSTIC FOR VAX-11/730 IDC MODULE

Product date: 28-APRIL-1982

Maintainer: BASE SYSTEMS DIAGNOSTIC ENGINEERING

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital or its affiliated companies.

Copyright (c) 1982 by Digital Equipment Corporation. All Rights Reserved.

The following are trademarks of Digital Equipment Corporation.

DEC
DECUS
UNIBUS

DECsystem-10
MASSBUS
VAX

DECSYSTEM-20
PDP
VMS

digital

Table of Contents

1.0	ABSTRACT	3
2.0	HARDWARE REQUIREMENTS	3
3.0	SOFTWARE REQUIREMENTS	3
4.0	PREREQUISITES	3
5.0	OPERATING INSTRUCTIONS	4
5.1	Event Flags	4
5.2	APT Setup	4
6.0	PROGRAM FUNCTIONAL DESCRIPTION	5
6.1	Program Overview	5
6.2	Program Size	5
6.3	Program Run Times	5
6.4	Fault Detection	5
6.5	Performance During Hardware Failures	6
6.6	Program Applications	6
6.7	Test Descriptions	6
7.0	MAINTENANCE HISTORY	6

1.0 ABSTRACT

This program is a micro-diagnostic designed to test the hardware on the IDC (Integrated Disk Controller) module of the VAX-11/730 processor.

2.0 HARDWARE REQUIREMENTS

This program will run with the minimum hardware configuration of a WCS, CPU, MCT, IDC, and array module.

Maximum main memory allowed is 5 one meg boards, for a total of 5 megabytes of main memory storage.

Other options may be installed without affecting the diagnostic's performance.

The IDC PROMS must contain V19.0 or later.

3.0 SOFTWARE REQUIREMENTS

This diagnostic must be run under the VAX-11/730 micro monitor (ENKAA) in a standalone environment. The micro-diagnostic is written into WCS RAM, wiping out any system micro-code that may have been resident. Some Diagnostic microcode is also located in the IDC control store. The micro-diagnostic monitor takes up the area where the console software is normally present.

4.0 PREREQUISITES

The WCS, DAP, and MCT modules are assumed to have been tested and known to be good before this diagnostic is run. The programs ENKBA through ENKBE, ENKCA through ENKCD will perform this testing.

5.0 OPERATING INSTRUCTIONS

This program may be executed by typing DI SE ENKCF after the micro-monitor has been loaded and the MIC> prompt has been printed at the terminal. Both the micro-monitor and this program are loaded from a TU58 cassette or downline loaded through the APT-RD port. See the OVERVIEW MANUAL for more information.

5.1 Event Flags

Not applicable

5.2 APT Setup

A Field Test Version (V0.0) of this diagnostic will not recognize when APT is present and will always size for the IDC module. If no IDC is found a message is printed and this diagnostic will not be run.

Version 1.0 of this diagnostic, will recognize when APT is present, and will report an error if the diagnostic does not find an IDC on the system.

The micro-monitor knows when APT is present by the position of the keyswitch and the state of a line connected to the APT-RD port. It sets a flag in the CPU Local Store which can be read by Version 1.0 and above of this diagnostic. When APT is present, the diagnostic will:

1. Size for the IDC. If no IDC is found, an error will be reported, and if there is an IDC, ENKCF will be run.

Under APT, the diagnostic will halt on error and report error information to APT via the mailbox in 8085 RAM.

The following describes the APT parameters needed to execute this diagnostic:

The Hardware Configuration Register must be loaded with the following information when running ENKCF: If an R80 is on Drive 0 and is powered up, Hardware Config Reg = 00000001 (01) If an R80 is on Drive 1 and is powered up, Hardware Config Reg = 00000010 (02) If an R80 is on Drive 2 and is powered up, Hardware Config Reg = 00000100 (04) If an R80 is on Drive 3 and is powered up, Hardware Config Reg = 00001000 (08) If there are no R80s powered up, Hardware Config Reg = 00000000 (00)

⋮

:55

:55
:55

:55
:55:55
:55
:55:55
:55
:55:56
:56
:56:56
:56
:56:56
:56
:56:56
:56
56:56
:56
57:57
:57
57:57
:57
57:57
:57
:57:57
:57
:57: 57
: 58

588

58
58
58

• 58

• 58
• 58

• 50
• 58
• 50

59
59

: 59
: 59

59

• 59
• 59

:59
:60:60
:60
:60:60:60

727	FIELD DEFINITIONS FOR MAIN MICRO WORD	VER 00.08
1733	MACRO DEFINITIONS	VER 00.02
2389	FIELD DEFINITIONS FOR DATA PATH CONTROL MICRO WORD	VER 00.01
2442	CONTROL ROM CONTENTS	VER 00.01
3191	DIAGNOSTIC MACROS AND DEFINITIONS	
3850	COMMON LS ASSIGNMENTS (TO REGISTERS)	
4010	Microinstruction Formats	
4393	Tables	
4524	2901 ALU Control Description	
4628	DIAGNOSTIC MACROS	
4656	TEST POINTERS	
4766	DIAGNOSTIC POINTERS	
4856	LS DATA SECTION	
5247	PROGRAM SPECIFIC DATA SECTION (LS 80-F7)	
5619	WCS SUBROUTINES FOR CONSOLE USE	
5620	GENERATE TRANSFER DATA	
5665	DEPOSIT MCT CSR ROUTINE	
5731	EXAMINE MCT CSR ROUTINE	
5818	DEPOSIT MCT TRANSLATION BUFFER ROUTINE	
5934	EXAMINE TRANSLATION BUFFER ROUTINE	
5997	DEPOSIT UNIBUS MAP ROUTINE	
6095	EXAMINE UNIBUS MAP ROUTINE	
6158	DEPOSIT MAIN MEMORY ROUTINE	
6216	EXAMINE MAIN MEMORY ROUTINE	
6278	EXAMINE IDC ROUTINES	
6344	DEPOSIT IDC ROUTINES	
6416	SAVE WR ROUTINE	
6467	RESTORE WR ROUTINE	
6518	WCS SUBROUTINES FOR CPU USE	
6519	ERROR ROUTINE	
6686	START TRANSFER ROUTINE	
6785	SUBROUTINES FOR IDC MICRODIAGNOSTIC	
6786	DISK DRIVE SIZING ROUTINE	
6920	INIT IDC AND GO TO DIAG.IDLE ROUTINE	
6996	WRITE..READ BYTE USING DISKLESS LOOP	
7223	CLEANUP ROUTINE	
7268	Test 1 Y BUS VERIFICATION TEST (M8388 IDC MODULE OR M8390 CPU MODULE)	
7393	Test 2 IDC SIZING TEST (M8388 IDC MODULE OR M8390 CPU MODULE)	
7566	Test 3 CRDY & XFER.REQ INIT TEST (M8388 IDC OR M8390 CPU MODULE)	
7734	Test 4 DISK ADDRESS REGISTER TEST (M8388 IDC MODULE)	
8007	Test 5 CONTROL STATUS REGISTER TEST (M8388 IDC MODULE)	
8309	Test 6 CRDY & IDC DIAGNOSTIC MICROCODE ACCESS TEST (M8388 IDC MODULE)	
8634	Test 7 CSR BITS <F0><F1><F2> BRANCH TEST (M8388 IDC MODULE)	
8931	Test 8 XFER.REQ-XFER GRANT TEST (M8388 IDC MODULE OR M8390 CPU MODULE)	
9139	Test 9 UINCREMENT DAR TEST (M8388 IDC MODULE)	
9373	Test A BEN0/CRDY AND BEN2/MAINT BRANCH TEST (M8388 IDC MODULE)	
9753	Test B CSR WRITE INHIBIT BIT <28> TEST (M8388 IDC MODULE)	
9864	Test C DRIVE READY BRANCH TEST (M8388 IDC MODULE)	
10125	Test D XFER.REQ BRANCH TEST (M8388 IDC MODULE)	
10432	Test E DISK DRIVE SIZING TEST (M8388 IDC MODULE)	
10761	Test F FIFO A ADRS CNTR MAX L BRANCH TEST 1 (M8388 IDC MODULE)	
11226	Test 10 FIFO A ADRS CNTR MAX L BRANCH TEST 2 (M8388 IDC MODULE)	
11721	Test 11 FIFO B ADRS CNTR MAX L BRANCH TEST 1 (M8388 IDC MODULE)	
12282	Test 12 FIFO B ADRS CNTR MAX L BRANCH TEST 2 (M8388 IDC MODULE)	
12863	Test 13 FIFO A ADRS CNTR OVFLW L BRANCH TEST 1 (M8388 IDC MODULE)	


```

1  TITLE 'ENKCF Micro-diagnostic for IDC module Rev 01.40'
2
3
4      COPYRIGHT (c) 1982 BY
5      DIGITAL EQUIPMENT CORPORATION, MAYNARD,
6      MASSACHUSETTS.  ALL RIGHTS RESERVED.
7
8  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
9  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE INCLUSION
10 OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR ANY OTHER COPIES THEREOF
11 MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON.  NO
12 TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY TRANSFERRED.
13
14 THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE, AND
15 SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.
16
17 DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
18 SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.
19
20
21 ++
22 FACILITY:  VAX-11/730 MICRO-DIAGNOSTIC.
23
24 ABSTRACT:  This micro-diagnostic tests the hardware of the IDC module
25             in the VAX-11/730 processor.
26
27 ENVIRONMENT:  ENKAA Micro-diagnostic Monitor
28
29 AUTHOR:      Vickie Hayes    24-MARCH-82    VERSION 1.1
30
31 MODIFIED BY:
32   REV. 1.1  VICKIE HAYES    24-MARCH-82
33
34   REV. 1.2  DAVID MAYO      28-APRIL-82
35             Changes to Tests 3C and 3D due to accomadate system clock
36             margining.
37
38   REV. 1.3  GEORGE HO       22-NOV-82
39             Addition of 3 error reports for fault insertion enhancement
40             in test 39.
41
42   REV. 1.4  PETER GREEN     15-MAR-83
43             MANUFACTURING PROBLEM W/SLOW CLOCK IDC NOT FOUND
44             (INTERMITTANT) BITS 25 & 27 BEING SET.  SET UP ERROR MASK
45             TO ONLY CHECK BITS <20:0> ON THE IDAR.
46
47 --
48 .nolist
49 .list
50 .NOCREF

```


:56 .RTOL
:57 .HEXADECIMAL
:58 .REVISION HISTOPY
:59
:60 VER 02.24 9-SEP-80 SCS
:61
:62 ADD THE BIC WR[] WITH WR[] TO Q MACRO.
:63
:64 VER 02.23 16-JUL-80 SCS
:65
:66 FIX MINC INSTRUCTION TO CALL INC INSTEAD OF DEC
:67
:68 VER 02.22 15-JUL-80 SCS
:69
:70 ADD SAVED.MDT TO LOCAL STORE.
:71
:72 VER 02.21 7-JUL-80 SCS
:73
:74 REMAP ALL OF THE CONSTANTS TO THE LOCAL STORE LOCATIONS
:75 REQUIRED BY THE 8085 TO BUILD THEM DURING POWER-UP.
:76
:77 THIS ALLOWS US TO RECOVER FROM POWER FAIL CORRECTLY.
:78
:79 THE CHANGES BASICALLY REQUIRED A RESHUFFLING OF THE CONSTANTS
:80 ACCORDING TO SIZE OF THE NUMBFR.
:81
:82 VER 02.20 13-JUN-80 SCS
:83
:84 FIX VALIDITY CHECKS TO ALLOW LEGAL SKIP CONDITIONS ON
:85 THE PAGE BOUNDARY CONSISTING OF:
:86
:87 RETURN
:88 RETURN+1
:89 RET.NOERR
:90 POP.USTACK
:91
:92 VER 02.19 04-JUN-80 SCS
:93
:94 ADD VALIDITY CONDITIONS TO THE SKIP AND JUMP CONTROL
:95 FIELDS TO CHECK FOR THE PAGE PROBLEM CREATED BY THE SEQUENCER. THIS
:96 LIMITATION ONLY ALLOWS CARRIES TO BE PROPAGATED FOR 10 BITS WHEN
:97 THE ADDRESS IS INCREMENTED BY TWO. THE NORMAL SEQUENCING OF INSTRUCTIONS
:98 CAN WALK ACROSS THESE 1K BOUNDARIES.
:99
:100 THIS REQUIRES ADDITION OF THE FOLLOWING VALIDITY CHECKS AND CHANGES
:101 TO THE SCTL AND JCTL FIELD'S VALIDITY CHECKS.
:102
:103 PAGE.PROB
:104 SKIP.PAGE.VAL
:105 JUMP.CHECK
:106 JUMP.PAGE.VAL
:107
:108 VER 02.18 02-JUN-80 SCS
:109
:110 REMOVE THE MACRO BIC WR[] WITH LS[] TO Q

: 781
 : 782
 : 783
 : 784
 : 785
 : 786
 : 787
 : 788
 : 789
 : 790
 : 791
 : 792
 : 793
 : 794
 : 795
 : 796
 : 797
 : 798
 : 799
 : 800
 : 801
 : 802
 : 803
 : 804
 : 805
 : 806
 : 807
 : 808
 : 809
 : 810
 : 811
 : 812
 : 813
 : 814
 : 815
 : 816
 : 817
 : 818
 : 819
 : 820
 : 821
 : 822
 : 823
 : 824
 : 825
 : 826
 : 827
 : 828
 : 829
 : 830
 : 831
 : 832
 : 833
 : 834
 : 835

```

:166 :
:167 :   ADD THE FOLLOWING CONSTANTS TO THE LOCAL STORE:
:168 :
:169 :   #40A10
:170 :
:171 :   ADD LOCATION IN LOCAL STORE:
:172 :
:173 :   MEMORY.SIZE
:174 :
:175 :   VER 02.12    23-FEB-80      SCS
:176 :
:177 :   ADD THE FOLLOWING NEW LABELS TO OLD LOCAL STORE
:178 :   LOCATIONS:
:179 :
:180 :   CONSOLE.COMMAND
:181 :   CONSOLE.MODIFIER
:182 :   CONSOLE.ADDRESS
:183 :   CONSOLE.STATUS
:184 :   CONSOLE.DATA
:185 :
:186 :   VER 02.11    11-FEB-80      SCS
:187 :
:188 :   UPDATE COMMENTS TO SWAP AND MEM FUNCTION.
:189 :
:190 :   VER 02.10    7-FEB-80      SCS
:191 :
:192 :   CHANGED CM.EXEC ADRS[] TO LOAD THE OS REGISTER.
:193 :
:194 :   VER 02.09    17-DEC-79      SCS
:195 :
:196 :   ADD THE CROSS REFERENCE LISTING TO OUTPUT.
:197 :
:198 :   BUILD THE FOLLOWING MEMORY CODES FOR THE MEM.FUNC FIELD:
:199 :
:200 :       WRITE.TB.STEP=11
:201 :       OCTA.READ.V.RCHK=1A
:202 :       READ.MAINT.UBS.DATA=1B
:203 :       OCTA.WR.V.WCHK=1D
:204 :       QUAD.READ.V.RCHK=1E
:205 :
:206 :   VER 02.08    12-NOV-79      SCS
:207 :
:208 :   CREATE BACKUP LOCATION FOR COMPATIBILITY MODE PC.
:209 :
:210 :       BACKUP.CM.PC
:211 :
:212 :   ADD THE FOLLOWING CONSTANTS:
:213 :
:214 :       #FF80
:215 :       #FFFF0
:216 :       #40211
:217 :       #66
:218 :
:219 :   VER 02.07    30-OCT-79      SCS
:220 :

```

ZZ-E
 :
 :
 :836
 :837
 :838
 :839
 :840
 :841
 :842
 :843
 :844
 :845
 :846
 :847
 :848
 :849
 :850
 :851
 :852
 :853
 :854
 :855
 :856
 :857
 :858
 :859
 :860
 :861
 :862
 :863
 :864
 :865
 :866
 :867
 :868
 :869
 :870
 :871
 :872
 :873
 :874
 :875
 :876
 :877
 :878
 :879
 :880
 :881
 :882
 :883
 :884
 :885
 :886
 :887
 :888
 :889
 :890

: 891
 : 892
 : 893
 : 894
 : 895
 : 896
 : 897
 : 898
 : 899
 : 900
 : 901
 : 902
 : 903
 : 904
 : 905
 : 906
 : 907
 : 908
 : 909
 : 910
 : 911
 : 912
 : 913
 : 914
 : 915
 : 916
 : 917
 : 918
 : 919
 : 920
 : 921
 : 922
 : 923
 : 924
 : 925
 : 926
 : 927
 : 928
 : 929
 : 930
 : 931
 : 932
 : 933
 : 934
 : 935
 : 936
 : 937
 : 938
 : 939
 : 940
 : 941
 : 942
 : 943
 : 944
 : 945

:276 : THIS IS THE VERSION WHICH PROVIDES COMPATIBILITY BETWEEN
 :277 : BREADBOARD ONE AND BREADBOARD TWO.

:278 :
 :279 :
 :280 : VALUES OF THE BIC FIELD ARE COMPLEMENTED.

:281 :
 :282 : MISC.2 FIELD VALUES ARE RE-ARRANGED.

:283 :
 :284 : MISC.4 FIELD REMOVED.

:285 :
 :286 : CHANGES TO THE FOLLOWING MACROS:

:287 :
 :288 : MOV WR[] TO WR[]
 :289 : MOV WR[] TO Q
 :290 : CLR Q
 :291 : CLR WR[]
 :292 : NEG WR[]
 :293 : INC WR[]
 :294 : DEC WR[]
 :295 : COM WR[]
 :296 : TST WR[]

:297 :
 :298 : ADD THE FOLLOWING MACROS

:299 :
 :300 : MCOM WR[] TO WR[]
 :301 : MINC WR[] TO WR[]
 :302 : MDEC WR[] TO WR[]

:303 :
 :304 : VER 01.23 11-SEP-79 SCS

:305 :
 :306 : ADD THE FOLLOWING CONSTANTS

:307 :
 :308 : #F0
 :309 : #A0
 :310 : #30
 :311 : #5F
 :312 : #5F40

:313 :
 :314 : UPDATE TO FIELD VALUE:

:315 :
 :316 : ASSERT.DA.AND.PASS.WR

:317 :
 :318 : RENAME NEW FIELDS:

:319 :
 :320 : MISC.WRL
 :321 : MISC.WRR

:322 :
 :323 : ADD NEW MACROS:

:324 :
 :325 : MOV FPA TO LS[]
 :326 : MOV PORT TO LS[]
 :327 : ASSERT.DA.AND.PASS.WR0
 :328 : ASSERT.DA.AND.PASS.WR1
 :329 : ADD Q TO WR[] XCHG TO LS[]

:330 :

```

331 : CHANGE MACROS:
332 :
333 : MISC[] WR[]
334 : MISC2[] WR[]
335 :
336 : REMOVE MACROS:
337 : (ALL PRIORITY ENCODE FUNCTIONS)
338 :
339 : RENAME MACRO TO
340 :
341 : LOOP.IF(NO INTERRUPT AND NO SYNC) ELSE SKIP.IF(SYNC)
342 :
343 : VER 01.22 02-AUG-79 SCS
344 :
345 : ADD HARDWARE INDEXES TO PORT REGISTERS.
346 :
347 : VER 01.21 31-JUL-79 SCS
348 :
349 : CHANGED CONSTANT #F03000 TO #F27000
350 : ADDED NEW MACROS
351 : MISC[] WR[]
352 : MISC2[] WR[]
353 :
354 : VER 01.20 20-JUL-79 SCS
355 :
356 : ADD FUNCTION
357 :
358 : MISC2[MASK.INTERRUPTS]
359 :
360 : VER 01.19 16-JUL-79 SCS
361 :
362 : ADD NEW MACRO
363 :
364 : MOV CM.IB.DATA TO OS
365 :
366 : ADD TRUE LSS TO SCTL FIELD.
367 : RENAME FALSE LSS TO N.CLR
368 : GEQ TO N.SET
369 : IN BOTH THE SCTL FIELD AND JCTL FIELD.
370 : THERE ARE NOW NO FREE SKIP CONDITIONS!!
371 :
372 : VER 01.18 13-JUL-79 SCS
373 :
374 : FIX ERROR WITH VALIDITY ON XCHG.BIT
375 :
376 : VER 01.17 07-JUL-79 SCS
377 :
378 : CHANGED #3A3000 TO #F03000
379 : INSERT ROTATE.BYTE.RIGHT INTO MEM.FUNC FIELD.
380 :
381 : ADD SYMMETRIC MACROS:
382 :
383 : SWAP WR[] WITH LSE[]
384 : ADD LSE[] PLUS WR[] TO Q
385 : BIS LSE[] WITH WR[] TO Q
  
```

```
:386 : AND LS[] WITH WR[] TO Q
:387 : XOR LS[] WITH WR[] TO Q
:388 :
:389 : ADD MNEG Q TO LS[]
:390 :
:391 : VER 01.16 08-JUN-79 SCS
:392 :
:393 : DEFINE THE FOLLOWING LOCATIONS:
:394 :
:395 : IE.TEMP1
:396 : IE.TEMP2
:397 : IE.TEMP3
:398 : IE.TEMP4
:399 :
:400 : #B020FF00
:401 : #C00000E0
:402 :
:403 : RENAME SKIP CONDITION IN SCTL FIELD TO:
:404 :
:405 : NO.FAST.INTERRUPT
:406 :
:407 : VER 01.15 16-MAY-79 SCS
:408 :
:409 : ADD LD.OS/LOAD.OS TO
:410 : DECODE.CM.OPC ADRS[]
:411 :
:412 : VER 01.14 14-MAY-79 SCS
:413 :
:414 : ADD A ZERO LOCATION TO LOCAL STORE.
:415 :
:416 : VER 01.13 08-MAY-79 SCS
:417 :
:418 : DEFINE TWO NEW FUNCTIONS FOR SI FIELD.
:419 :
:420 : UMUL.SHF
:421 : SHF.WR.Q
:422 :
:423 : ADD THE FOLLOWING MACROS:
:424 :
:425 : UNSIGNED.MUL WR[] TO WR[].Q
:426 : SHR WR[]
:427 : SHL WR[]
:428 : SHRC WR[].Q
:429 : SHLC WR[].Q
:430 :
:431 : VER 01.12 01-MAY-79 SCS
:432 :
:433 : INSERT LS[CONSOLE.CSR.IES]
:434 :
:435 : VER 01.11 19-APR-79 SCS
:436 :
:437 : ADDITION OF EQL PNEUMONIC TO THE JCTL FIELD. CHANGE
:438 : TO DPCROM TO FIX THE FOLLOWING MACROS:
:439 :
:440 : ADD Q PLUS LS[] TO WR[]
```



```
:441 : COND.ADD&SHIFT WR[] TO WR[].Q
:442 :
:443 : VER 01.10 10-APR-79 SCS
:444 :
:445 : ADDITION OF .SET/UPC=000 TO THE DEFINITION FILE SO THAT ALL
:446 : MICRO ASSEMBLERS CAN USE .CHANGE/UPC=<.+1> IN THEIR CODE.
:447 :
:448 : VER 01.09 04-APR-79 SCS
:449 :
:450 : FIX DECODE.CM.DST ADRS[]
:451 :
:452 : ADD SAVE.CM.PC WRO
:453 :
:454 : UPDATE MOV IB.DATA TO OS
:455 :
:456 : SWITCH JCTL FIELDS
:457 : NO.JUMP.TST
:458 : JUMP.VALID
:459 :
:460 : VER 01.08 30-MAR-79 SCS
:461 :
:462 : ADD IN OS.BAK
:463 :
:464 : ADD IN TST Q
:465 :
:466 : VER 01.07 21-MAR-79 SCS
:467 :
:468 : REMOVE THE FOLLOWING IMPOSSIBLE MACROS:
:469 :
:470 : BIC Q TO LS[]
:471 : BIC WR[] TO LS[]
:472 : BIC Q TO WR[]
:473 : BIC WR[] WITH LS[] TO Q
:474 :
:475 : UPDATE THE FOLLOWING MACROS TO WORK:
:476 :
:477 : OPR Q TO WR[]
:478 :
:479 : ADD THE FOLLOWING MACRO:
:480 :
:481 : ADD Q PLUS LS[] TO WR[]
:482 :
:483 : VER 01.06 19-MAR-79 SCS
:484 :
:485 : ADD NEW LOCATIONS TO LOCAL STORE FOR MEMORY MANAGEMENT
:486 : ADD TWO NEW TEMPORARY LOCATIONS.
:487 : FIX UP DPCROM FOR
:488 :
:489 : PRI.ENC WRO R.TO.L TO OS AND CLEAR BIT
:490 :
:491 : ADD THE FOLLOWING CONSTANTS:
:492 :
:493 : #FF
:494 : #FFFFFFF
:495 : #3A3000
```

```

:496 : #7FFFFFF00
:497 : #3FFFFFF00
:498 : #3B
:499 : #3F
:500 : #3000
:501 : #FFFFFF000
:502 : #FFEO
:503 : #FFF
:504 : #0FFF0000
:505 : #FFFFFF8F
:506 :
:507 : ADD THE FOLLOWING MACROS:
:508 :
:509 : ADD Q PLUS WR[] TO LS[]
:510 : SUB Q FROM WR[] TO LS[]
:511 :
:512 : FIX UP ERROR IN MACRO:
:513 :
:514 : MOV XWR[] TO Q
:515 : CMP Q WITH WR[]
:516 :
:517 : VER 01.05 13-MAR-79 SCS
:518 :
:519 : FIX MEM.FUNC FIELD TO MATCH HARDWARE.
:520 :
:521 : FIX ERROR IN PRI.ENC MACRO IN DATA PATH CONTROL ROM.
:522 :
:523 : FIX ERROR IN MISC.2 FIELD
:524 :
:525 : VER 01.04 07-MAR-79 SCS
:526 :
:527 : FIX ERROR IN THE LABEL IN DATA PATCH CONTROL ROM
:528 : CORRESPONDING TO THE FOLLOWING MACRO:
:529 :
:530 : BIS WR[] WITH LS[] TO Q
:531 :
:532 : CHANGE TST WR[] MACRO TO USE NEW DATA PATH CONTROL ROM FUNCTION
:533 : WHICH WILL CLEAR THE V AND C CONDITION CODES IF DT(SIZE)&SET.ALU.CC
:534 : IS INVOKED.
:535 :
:536 : INSERT NEW MACROS FOR COMPATIBILITY MODE DECODE MACROS:
:537 :
:538 : JMP.TO []
:539 : JSR.TO []
:540 :
:541 : FIX ERROR IN:
:542 :
:543 : DECODE.CM.SINGLE ADRS[]
:544 :
:545 : VER 01.03 05-MAR-79 SCS
:546 :
:547 : FIX ERROR IN THE FOLOWING MACROS:
:548 :
:549 : SET.STATE.ZERO&CLR.STATE.ONE
:550 : SET.STATE.ONE&CLR.STATE.ZERO
  
```

```

:551 :
:552 :   INSERT CM.PC LOCATION OF THE LOCAL STORE.
:553 :
:554 :   INSERT THE FOLLOWING MACROS:
:555 :
:556 :   LOOP.IF(NO INTERRUPT AND NO SYNC)
:557 :
:558 :   VER 01.02      01-MAR-79      SCS
:559 :
:560 :   IMPLEMENT NEW PRIORITY ENCODE FUNCTIONS AND DOCUMENT:
:561 :
:562 :   PRI.ENC WRO R.TO.L TO OS
:563 :   PRI.ENC WRO L.TO.R TO OS
:564 :   PRI.ENC WRO R.TO.L TO OS AND CLEAR BIT
:565 :   PRI.ENC WRO L.TO.R TO OS AND CLEAR BIT
:566 :
:567 :   COMPLEMENT ALL FIELD VALUES OF THE MISC.1 FIELD.
:568 :   ADD VAR FIELD VALUE FOR WR[5].
:569 :   PUT THE FOLLOWING LOCATIONS INTO THE HIGH HALF OF LS:
:570 :       POBR
:571 :       POLR
:572 :       P1BR
:573 :       P1LR
:574 :       SBR
:575 :       SLR
:576 :
:577 :   RESERVE LOCATIONS A0-BF FOR THE PORT. (PORT0-PORT31)
:578 :   REMOVE TEMPORARY LOCATIONS T22-T28 LEAVING T0-T15 AS
:579 :   THE TEMPORARY LOCATIONS AVAILABLE.
:580 :   INSERT .PAGE 'S FOR READABILITY.
:581 :
:582 :   VER 01.01      01-MAR-79      SCS
:583 :
:584 :   FIX THE SUB LS[] FROM WR[] TO Q MACRO.
:585 :
:586 :   VER 01.00      23-FEB-79      SCS
:587 :
:588 :   THIS VERSION REPRESENTS THE CHANGES TO MATCH THE HARDWARE CHANGES
:589 :   TO BE IMPLEMENTED BEFORE MARCH 1,1979.
:590 :
:591 :   DP.SMALL AND XCHG.BIT FIELDS CREATED TO HELP EXCHANGE FUNCTION.
:592 :   SCTL VALUES CHANGED.
:593 :   JCTL VALUES CHANGED.
:594 :   IFUNC VALUES CHANGED AND CREATION OF OPC.SPEC FIELD.
:595 :   MISC.PORT FIELD CREATED.
:596 :   MISC.1 AND MISC.2 VALUES CHANGED.
:597 :   MISC.4 VALUES CREATED.
:598 :   MEM.FUNC VALUES CHANGED.
:599 :   CREATION OF MISC2[] MACRO AND UPDATE OF ALL MISCELLANEOUS MACROS.
:600 :   RESHUFFLING OF CONTROL ROM TO MATCH HARDWARE CHANGES AND
:601 :   ADDITION OF ENTRIES TO CONTROL ROM FOR NEW MACROS.
:602 :
:603 :   ADDITIONAL COMMENTS TO EXPLAIN OPERATION OF ALU CC'S FOR
:604 :   ALL OPERATIONS.
:605 :
  
```

```

:606 :      ADDITION OF THE FOLLOWING MACROS:
:607 :
:608 :          XCHG
:609 :          SWAP LS[] WITH WR[]
:610 :          MOV MEM.DATA TO WR[] XCHG TO LS[]
:611 :          SET.BACKUP.MASK.VALID
:612 :          PRIORITY ENCODE WRO R.TO.L TO OS
:613 :          PRIORITY ENCODE WRO L.TO.R TO OS
:614 :
:615 :      CONVERT ALL Q.WR[] MACROS TO WR[].Q
:616 :      ADD TIMER ENTRY TO LOCAL STORE.
:617 :
:618 :      VER 00.17      13-FEB-79      SCS
:619 :
:620 :          SWITCH READ CHECK AND WRITE CHECK FUNCTIONS FOR THE MEMORY.
:621 :
:622 :      VER 00.16      12-FEB-79      SCS
:623 :
:624 :          CHANGED MDT VALUES TO CORRECT ONES.
:625 :          CHANGE MEMORY FUNCTION VALUES TO MATCH HARDWARE.
:626 :
:627 :      VER 00.15      31-JAN-79      SCS
:628 :
:629 :          SPLIT SOURCE OF DEF.MIC INTO THREE FILES:
:630 :          DEFPREFIX.MIC
:631 :          DEFDEF.MIC
:632 :          DPCROM.MIC
:633 :
:634 :          ADD CORRECT VALUES AND NAMES OF ALL MEMORY FUNCTIONS.
:635 :
:636 :      VER 00.14      29-JAN-79      SCS
:637 :
:638 :          UPDATE COMMENTS ABOUT PSL STATE.
:639 :
:640 :      VER 00.13      15-JAN-79      SCS
:641 :
:642 :          ADD MACROS FOR RORC Q.WR[] AND ROLC Q.WR[]
:643 :
:644 :      VER 00.12      12-JAN-79      SCS
:645 :
:646 :          FIX SHL2, ASHRC, ASHLC
:647 :          DEFINED DEBUGGER LOCAL STORE LOCATIONS
:648 :          DEFINED ALL MEMORY FUNCTION NAMES (VALUES NOT FINAL)
:649 :
:650 :
:651 :      VER 00.00
:652 :
:653 :          INITIAL RELEASE TO LIBRARY
:654 :
:655 :      VER 00.01
:656 :
:657 :          REDEFINE OPCODE BITS PER HARDWARE IMPLEMENTATION
:658 :
:659 :      VER 00.02      19-SEPT-78
:660 :

```

:661 : ADD MACROS FOR STATE REGISTER CONTROL
:662 : UPDATE SKIP CONTROL FIELD FOR NEW ITERATION CONTROL FUNCTIONS
:663 : ADD MISC FUNCTIONS FOR MASKING TP AND CONSOLE HALT
:664 : ADD MISC FUNCTION FOR MEMORY RESET
:665 : ASSIGN LOCAL STORE ADDRESS FOR SOFT PSL
:666 : ASSIGN MEMORY MANAGEMENT REGISTERS
:667 : ASSIGN EXTENDED TEMP LOCATIONS FOR STACK POINTERS CONTROL BLOCK BASE
:668 : ADD NEW CONSTANTS
:669 :
:670 : VER 00.03 22-SEPT-78
:671 :
:672 : ADD EXTENDED B ADDRESS MACRO
:673 : ADD STATUS REGISTER NAMES
:674 : ADD ALTERNATE NAMES FOR CONSTANTS
:675 : FIX MISC [] MACRO
:676 : ADD CONSTANT 1F
:677 : ADD ABILITY TO SKIP ON BOTH POLARITIES OF THE STATE REGISTER
:678 :
:679 : VER 00.04 4-OCT-78
:680 :
:681 : UPDATE 2901 CONTROL PER HARDWARE IMPLSPEC
:682 : ADD CS PARITY GENERATION
:683 : CHANGE ADDRESS OF MACRO PC IN LS
:684 : UPDATE SKIP AND JUMP CONTROL ASSIGNMENTS
:685 : ADD CONSTANTS
:686 : FIX XWR MOVES
:687 :
:688 : VER 00.05 11-OCT-78
:689 :
:690 : REVISE SYNTAX FROM DESIGN REVIEW
:691 :
:692 : VER 00.06 13-OCT-78
:693 :
:694 : FIX MEM.REQ MACRO
:695 : ADD XD CONSTANTS
:696 :
:697 : VER 00.07 16-OCT-78
:698 :
:699 : FIX SCTL AND JCTL CODES FOR Z AND C BITS
:700 :
:701 : VER 00.08 20-OCT-78
:702 :
:703 : ADD MEMORY FUNCTION FOR READ CONTROL REGISTER
:704 : FIX LOOP CONTROL MACRO FOR GEQU
:705 : ADD BACKUP GPR ADDRESSES
:706 : ADD CONSTANTS
:707 :
:708 : VER 00.09 27-OCT-78
:709 :
:710 : UPDATE CONTROL FIELDS TO REFLECT NEW HARDWARE SPEC
:711 : ADD SHIFT FIELD CODES
:712 :
:713 : VER 00.10 20-DEC-78
:714 :
:715 : Add shift and rotate macros


```

:716      : Change all CLR macros to 'R.AND.S' from R.XNOR.S
:717      :
:718      : VER 00.11      4-JAN-79
:719      :
:720      : Eliminate redundant macros for shift & rotate
:721      : Add MNEG WR[] WR[]
:722      : Fill in gaps between groups of macros for ROM sake
:723      :
:724      :
:725      :

```

```

:726 .PAGE
:727 .TOC 'FIELD DEFINITIONS FOR MAIN MICRO WORD VER 00.08''
:728 .SET/UPC=<000>
:729
:730 ; THE FOLLOWING EXPRESSIONS ARE VALIDITY CHECKS FOR FIELD COMBINATIONS
:731
:732 .SET/A.VAL=<.CASE[<OPC2/>]OF[0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0]>
:733 .SET/B.VAL=<.CASE[<OPC2/>]OF[0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1]>
:734 .SET/D.VAL=<.CASE[<OPC2/>]OF[0,0,0,1,0,0,0,0,1,1,1,1,1,1,1,1]>
:735 .SET/XD.VAL=<.EQL[<OPC1/>,<OPC1/MOVE>]>
:736 .SET/CC.VAL=<.CASE[<OPC2/>]OF[0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1]>
:737 .SET/DT.VAL=<.EQL[<OPC2/>,<OPC2/MEM.REQ>]>
:738 .SET/SCTL.VAL=<.CASE[<OPC2/>]OF[0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1]>
:739 .SET/JCTL.VAL=<.CASE[<OPC2/>]OF[1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0]>
:740 .SET/JUMP.VAL=<.EQL[<OPC2/>,<OPC2/JUMP>]>
:741 .SET/DECODE.VAL=<.EQL[<OPC2/>,<OPC2/DECODE>]>
:742 .SET/MISC.VAL=<.EQL[<OPC2/>,<OPC2/MISC>]>
:743 .SET/DP.VAL=<.EQL[<OPC/>,<OPC/BASIC>]>
:744
:745 ; THE FOLLOWING VALIDITY CHECKS ARE USE FOR THE PAGE BOUNDARY PROBLEM.
:746
:747 .SET/PAGE.PROB=<.EQL[<.AND[<7FF>,<.>]>,<7FE>]>
:748 .SET/SKIP.LEGAL2=<.OR[<.EQL[<SCTL/>,<SCTL/RET.NOERR>]>,<.EQL[<SCTL/>,<SCTL/POP.USTACK>]>]>
:749
:750 .SET/SKIP.LEGAL=<.OR[<.EQL[<SCTL/>,<SCTL/NO.SKIP>]>,<.EQL[<SCTL/>,<SCTL/RETURN>]>,<.EQL[<SCTL/>,<SCTL/RETURN+1>]>,<SKIP.LEGAL2>]>
:751
:752 .SET/SKIP.PAGE.VAL=<.CASE[<PAGE.PROB>]OF[1,<SKIP.LEGAL>]>
:753 .SET/JUMP.CHECK=<.OR[<.EQL[<JCTL/>,<JCTL/JSR>]>,<.EQL[<JCTL/>,<JCTL/JSR.VALID>]>]>
:754
:755 .SET/JUMP.PAGE.VAL=<.CASE[<PAGE.PROB>]OF[1,<.XOR[1,<JUMP.CHECK>]>]>
:756
:757
:758 ; MICRO INSTRUCTION OPCODE DEFINITIONS
:759
:760 OPC/=<22>,.DEFAULT=<OPC/BASIC> ; MICRO OPCODE FIELD FOR SINGLE BIT OPCODE.
:761
:762 BASIC=1 ; OPCODE FOR 'BASIC' MICRO INSTRUCTION
:763
:764 OPC1/=<22:20> ; MICRO OPCODE FIELD FOR THREE BIT OPCODES.
:765
:766 EXTENDED=2 ; OPCODE FOR 'EXTENDED' MICRO INSTRUCTION
:767 MOVE=3 ; OPCODE FOR 'MOVE' MICRO INSTRUCTION
:768
:769 OPC2/=<22:19> ; MICRO OPCODE FIELD FOR FOUR BIT OPCODES
:770
:771 MEM.REQ=3 ; OPCODE FOR 'MEM.REQ' MICRO INSTRUCTION
:772 MISC=2 ; OPCODE FOR 'MISC' MICRO INSTRUCTION
:773 JUMP=1 ; OPCODE FOR 'JUMP' MICRO INSTRUCTION
:774 DECODE=0 ; OPCODE FOR 'DECODE' MICRO INSTRUCTION
:775
:776
:777 ; THE FOLLOWING FOUR FIELDS ARE RELATED TO ADDRESSING
:778 ; VARIOUS RAM MEMORIES WITHIN THE DATA PATH.
:779
:780 ; THE A.ADRS REFERS TO THE 2901 INTERNAL RAM 'A-PORT'

```

```

:781 : THE B.ADRS REFERS TO THE 2901 INTERNAL RAM 'B-PORT'
:782 : THE XB.ADRS REFERS TO THE 2901 INTERNAL RAM 'B-PORT'
:783 : THE D.ADRS REFERS TO THE LOWER 128 LOCATIONS OF THE LOCAL STORE RAM
:784 : THE XD.ADRS REFERS TO ALL 256 LOCATIONS OF THE LOCAL STORE RAM
:785 :
:786 : THE VALIDITY STATEMENTS ALLOW USAGE OF THESE FIELDS IN THE FOLLOWING MANNER
:787 :
:788 :     A.ADRS      EXTENDED MICRO INSTRUCTION
:789 :
:790 :     B.ADRS      BASIC MICRO INSTRUCTION
:791 :                 EXTENDED MICRO INSTRUCTION
:792 :                 MOVE MICRO INSTRUCTION
:793 :                 DECODE.IS MICRO INSTRUCTION
:794 :
:795 :     XB.ADRS      MOVE XWR MICRO INSTRUCTION
:796 :
:797 :     D.ADRS      BASIC MICRO INSTRUCTION
:798 :                 MEM.REQ MICRO INSTRUCTION
:799 :
:800 :     XD.ADRS      MOVE MICRO INSTRUCTION
:801 :
:802 : A.ADRS/= <10:9> ,.VALIDITY=<A.VAL>
:803 :
:804 :     MASK=3      ; REGISTER BACKUP MASK
:805 :
:806 : B.ADRS/= <8:7> ,.VALIDITY=<B.VAL> ,.DEFAULT=0
:807 :
:808 :     MASK=3      ; REGISTER BACKUP MASK
:809 :
:810 : ; THIS ADDRESS FIELD WILL CONTAIN THE LOW TWO BITS OF THE 2901 B ADDRESS
:811 : ; THE THIRD BIT WILL BE FORCED BY HARDWARE.
:812 :
:813 : XB.ADRS/= <8:7> ,.VALIDITY=<.EQL[<OPC1/>,<OPC1/MOVE>]>
:814 :
:815 :     BPC=0        ; ADDRESS OF BEGINNING PC      WR[4]
:816 :     VA=1         ; ADDRESS OF MEMORY REFERENCE WR[5]
:817 :     VAR=1        ; ADDRESS OF LAST MEMORY REFERENCE. (WR[5] IN 2901).
:818 :
:819 : ; D.ADRS/= <15:9> ,.VALIDITY=<D.VAL> ,.DEFAULT=0
:820 :
:821 :
:822 :     SAVED.2ND.REQUEST=0 ; MM SAVED STATE FOR REQUEST.
:823 :     T1=1          ; TEMP LOCATION 1
:824 :     T2=2          ; TEMP LOCATION 2
:825 :     T3=3          ; TEMP LOCATION 3
:826 :     T4=4          ; TEMP LOCATION 4
:827 :     T5=5          ; TEMP LOCATION 5
:828 :     T6=6          ; TEMP LOCATION 6
:829 :     T7=7          ; TEMP LOCATION 7
:830 :     T8=8          ; TEMP LOCATION 8
:831 :     T9=9          ; TEMP LOCATION 9
:832 :     T10=0A        ; TEMP LOCATION 10
:833 :     BACKUP.CM.PC=0A ; COMPATIBILITY MODE KEEPS ORIGINAL PC HERE.
:834 :
:835 :     T11=0B        ; TEMP LOCATION 11

```

```

:836 :      T12=0C      : TEMP LOCATION 12
:837 :      T13=0D      : TEMP LOCATION 13
:838 :      T14=0E      : TEMP LOCATION 14
:839 :      T15=0F      : TEMP LOCATION 15
:840 :      PC=10       : MACRO PROGRAM COUNTER
:841 :      WR.BACKUP=11 : BACKUP WORKING REGISTER FOR MOV MEM.DATA TO WR[]
:842 :      SAVED.VAR=12 : MM SAVED STATE FOR VAR
:843 :      SAVED.DATA=13 : MM SAVED STATE FOR DATA
:844 :      SAVED.PTE.ADDRESS=14 : MM SAVED STATE FOR PAGE TABLE ENTRIES ADDRESS
:845 :      SAVED.PTE=15 : MM SAVED STATE FOR PAGE TABLE ENTRIES
:846 :      T16=16      : TEMP LOCATION 16
:847 :      T17=17      : TEMP LOCATION 17
:848 :      IE.TEMP4=18  : INTERRUPTS AND EXCEPTIONS TEMPORARY LOCATION FOUR.
:849 :      #FFFFFF00=19 : CONSTANT
:850 :      #FFFFFF00=1A : CONSTANT
:851 :      #FFFFFFFC=1B : CONSTANT
:852 :      #FFFFFFF=1C  : CONSTANT
:853 :
:854 :      THIS LOCATION CONTAINS THE SOFT COPY OF THE PSL. WHENEVER THE
:855 :      HARDWARE VERSION OF THE PSL IS CHANGED (PSL.HW), THEN THIS
:856 :      CHANGE IS ALSO REPORTED HERE BY THE MICROCODE. ALL BITS OF
:857 :      THE PSL ARE REALLY LIKE THOSE IN THIS WORD EXCEPT FOR THE
:858 :      CONDITION CODES WHICH MUST BE OBTAINED FROM PSL.CC
:859 :
:860 :      PSL=1D       : SOFT COPY OF VAX PSL
:861 :      VAR=1E       : VIRTUAL ADDRESS REGISTER
:862 :      VAR2=1F      : VIRTUAL ADDRESS REGISTER
:863 :
:864 :
:865 :      #1=20        : MASK 00000001 (HEX)
:866 :      #2=21        : MASK 00000002
:867 :      #4=22        : MASK 00000004
:868 :      #8=23        : MASK 00000008
:869 :      #10=24       : MASK 00000010
:870 :      #20=25       : MASK 00000020
:871 :      #40=26       : MASK 00000040
:872 :      #80=27       : MASK 00000080
:873 :      #100=28      : MASK 00000100
:874 :      #200=29      : MASK 00000200
:875 :      #400=2A      : MASK 00000400
:876 :      #800=2B      : MASK 00000800
:877 :      #1000=2C     : MASK 00001000
:878 :      #2000=2D     : MASK 00002000
:879 :      #4000=2E     : MASK 00004000
:880 :      #8000=2F     : MASK 00008000
:881 :      #10000=30    : MASK 00010000
:882 :      #20000=31    : MASK 00020000
:883 :      #40000=32    : MASK 00040000
:884 :      #80000=33    : MASK 00080000
:885 :      #100000=34   : MASK 00100000
:886 :      #200000=35   : MASK 00200000
:887 :      #400000=36   : MASK 00400000
:888 :      #800000=37   : MASK 00800000
:889 :      #1000000=38  : MASK 01000000
:890 :      #2000000=39  : MASK 02000000

```

22-ENKCF-1.4	:	ENKCF.MIC	FIELD DEFINITIONS F15-MAR-83	N 2	Fiche 1 Frame N2	Sequence 26			22-E
:	:	ENKCF.MCR	MICRO2 1M(01)	15-MAR-83 11:46:22	ENKCF Micro-diagnostic for IDC module	Rev 01.40	Page	20	:
:	:	ENKCF.MIC	FIELD DEFINITIONS FOR MAIN MICRO WORD		VER 00.08				:

:891	:	#4000000=3A	:	MASK 04000000	:156
:892	:	#8000000=3B	:	MASK 08000000	:156
:893	:	#10000000=3C	:	MASK 10000000	:156
:894	:	#20000000=3D	:	MASK 20000000	:156
:895	:	#40000000=3E	:	MASK 40000000	:156
:896	:	#80000000=3F	:	MASK 80000000	:157
:897	:		:		:157
:898	:	BIT0=20	:	MASK 00000001	:157
:899	:	BIT1=21	:	MASK 00000002	:157
:900	:	BIT2=22	:	MASK 00000004	:157
:901	:	BIT3=23	:	MASK 00000008	:157
:902	:	BIT4=24	:	MASK 00000010	:157
:903	:	BIT5=25	:	MASK 00000020	:157
:904	:	BIT6=26	:	MASK 00000040	:157
:905	:	BIT7=27	:	MASK 00000080	:157
:906	:	BIT8=28	:	MASK 00000100	:158
:907	:	BIT9=29	:	MASK 00000200	:158
:908	:	BIT10=2A	:	MASK 00000400	:158
:909	:	BIT11=2B	:	MASK 00000800	:158
:910	:	BIT12=2C	:	MASK 00001000	:158
:911	:	BIT13=2D	:	MASK 00002000	:158
:912	:	BIT14=2E	:	MASK 00004000	:158
:913	:	BIT15=2F	:	MASK 00008000	:158
:914	:	BIT16=30	:	MASK 00010000	
:915	:	BIT17=31	:	MASK 00020000	
:916	:	BIT18=32	:	MASK 00040000	
:917	:	BIT19=33	:	MASK 00080000	
:918	:	BIT20=34	:	MASK 00100000	
:919	:	BIT21=35	:	MASK 00200000	
:920	:	BIT22=36	:	MASK 00400000	
:921	:	BIT23=37	:	MASK 00800000	
:922	:	BIT24=38	:	MASK 01000000	
:923	:	BIT25=39	:	MASK 02000000	
:924	:	BIT26=3A	:	MASK 04000000	
:925	:	BIT27=3B	:	MASK 08000000	
:926	:	BIT28=3C	:	MASK 10000000	
:927	:	BIT29=3D	:	MASK 20000000	
:928	:	BIT30=3E	:	MASK 40000000	
:929	:	BIT31=3F	:	MASK 80000000	
:930	:		:		
:931	:	GPR0=40	:	GPR 0	
:932	:	GPR1=41	:	GPR 1	
:933	:	GPR2=42	:	GPR 2	
:934	:	GPR3=43	:	GPR 3	
:935	:	GPR4=44	:	GPR 4	
:936	:	GPR5=45	:	GPR 5	
:937	:	GPR6=46	:	GPR 6	
:938	:	CM.SP=46	:	IN COMPATIBILITY THIS IS THE STACK POINTER.	
:939	:	GPR7=47	:	GPR 7	
:940	:	CM.PC=47	:	IN COMPATIBILITY MODE THIS IS THE PC.	
:941	:	GPR8=48	:	GPR 8	
:942	:	GPR9=49	:	GPR 9	
:943	:	GPR10=4A	:	GPR 10	
:944	:	GPR11=4B	:	GPR 11	
:945	:	GPR12=4C	:	GPR 12	

:946	:	AP=4C	:	ARGUMENT POINTER
:947	:	GPR13=4D	:	GPR 13
:948	:	FP=4D	:	FRAME POINTER
:949	:	SP=4E	:	GPR 14
:950	:	T0=4F	:	TEMP LOCATION 0
:951	:		:	
:952	:	ZERO=50	:	ZERO CONSTANT
:953	:	#3=51	:	CONSTANT
:954	:	#5=52	:	CONSTANT
:955	:	#6=53	:	CONSTANT
:956	:	#7=54	:	CONSTANT
:957	:	#9=55	:	CONSTANT
:958	:	#B=56	:	CONSTANT
:959	:	#C=57	:	CONSTANT
:960	:	#D=58	:	CONSTANT
:961	:	#E=59	:	CONSTANT
:962	:	#F=5A	:	CONSTANT
:963	:	#13=5B	:	CONSTANT
:964	:	#1F=5C	:	CONSTANT
:965	:	#34=5D	:	CONSTANT
:966	:	#3B=5E	:	CONSTANT
:967	:	#3F=5F	:	CONSTANT
:968	:	#4B=60	:	CONSTANT
:969	:	#66=61	:	CONSTANT
:970	:	#A0=62	:	CONSTANT
:971	:	#F0=63	:	CONSTANT
:972	:	#FF=64	:	CONSTANT
:973	:	#1FF=65	:	CONSTANT
:974	:	#FFF=66	:	CONSTANT
:975	:	#2001=67	:	CONSTANT
:976	:	#3000=68	:	CONSTANT
:977	:	#7F80=69	:	CONSTANT
:978	:	#C000=6A	:	CONSTANT
:979	:	#FFE0=6B	:	CONSTANT
:980	:	#FFFF=6C	:	CONSTANT
:981	:	#1F0000=6D	:	CONSTANT
:982	:	#200001=6E	:	CONSTANT
:983	:	#F27000=6F	:	CONSTANT
:984	:	#3000000=70	:	CONSTANT
:985	:	#41F0000=71	:	CONSTANT
:986	:	#7000000=72	:	CONSTANT
:987	:	#3FFFFFF00=73	:	CONSTANT
:988	:	#7F800000=74	:	CONSTANT
:989	:	#7FFFFFF00=75	:	CONSTANT
:990	:	#7FFFFFF0E=76	:	CONSTANT
:991	:	#BF800001=77	:	CONSTANT
:992	:		:	
:993	:	GPR.OS=78	:	HARDWARE INDEX TO GPRS
:994	:	BIT.OS(3-0)=79	:	16 LOCATION HARDWARE INDEX TO BIT MASKS
:995	:	BIT.OS(4-0)=7B	:	32 LOCATION HARDWARE INDEX TO BIT MASKS
:996	:	OS=7C	:	OS REGISTER
:997	:	DEC.CON=7D	:	DECIMAL CARRIES
:998	:	SIZE=7E	:	SIZE REGISTER
:999	:	MDT= 7E	:	MEMORY REFERENCE DATA SIZE
:1000	:	INTERRUPT.VEC=7E	:	HARDWARE INTERRUPT VECTOR


```

:1001 :
:1002 : THIS WORD IS THE HARDWARE CONDITION CODES. THE CC'S CAN BE READ
:1003 : OR ARE WRITTEN HERE. NO OTHER BITS IN THE PSL ARE EFFECTED.
:1004 :
:1005 : PSL.CC=7F ; PSL CONDITION CODES
:1006 :
:1007 : XD.ADRS/=<16:9>,.VALIDITY=<XD.VAL>
:1008 :
:1009 : SAVED.2ND.REQUEST=0 ; MM SAVED STATE FOR REQUEST.
:1010 : T1=1 ; TEMP LOCATION 1
:1011 : T2=2 ; TEMP LOCATION 2
:1012 : T3=3 ; TEMP LOCATION 3
:1013 : T4=4 ; TEMP LOCATION 4
:1014 : T5=5 ; TEMP LOCATION 5
:1015 : T6=6 ; TEMP LOCATION 6
:1016 : T7=7 ; TEMP LOCATION 7
:1017 : T8=8 ; TEMP LOCATION 8
:1018 : T9=9 ; TEMP LOCATION 9
:1019 : T10=0A ; TEMP LOCATION 10
:1020 : BACKUP.CM.PC=0A ; COMPATIBILITY MODE KEEPS ORIGINAL PC HERE.
:1021 : T11=0B ; TEMP LOCATION 11
:1022 : T12=0C ; TEMP LOCATION 12
:1023 : T13=0D ; TEMP LOCATION 13
:1024 : T14=0E ; TEMP LOCATION 14
:1025 : T15=0F ; TEMP LOCATION 15
:1026 : PC=10 ; MACRO INSTRUCTION PC
:1027 : WR.BACKUP=11 ; BACKUP WORKING REGISTER FOR MOV MEM.DATA TO WR[].
:1028 : SAVED.VAR=12 ; MM SAVED VAR LOCATION
:1029 : SAVED.DATA=13 ; MM SAVED DATA LOCATION
:1030 : SAVED.PTE.ADDRESS=14 ; MM SAVED PAGE TABLE ENTRY ADDRESS
:1031 : SAVED.PTE=15 ; MM SAVED PAGE TABLE ENTRY
:1032 : T16=16 ; TEMP LOCATION 16
:1033 : T17=17 ; TEMP LOCATION 17
:1034 : IE.TEMP4=18 ; INTERRUPTS AND EXCEPTIONS TEMPORARY LOCATION FOUR.
:1035 : #FFFFFF00=19 ; CONSTANT
:1036 : #FFFFFF00=1A ; CONSTANT
:1037 : #FFFFFFFC=1B ; CONSTANT
:1038 : #FFFFFFFF=1C ; CONSTANT
:1039 :
:1040 : THIS LOCATION CONTAINS THE SOFT COPY OF THE PSL. WHENEVER THE HARDWARE
:1041 : VERSION OF THE PSL IS CHANGED (PSL.HW) THEN THIS CHANGE IS
:1042 : REPORTED HERE. ALL BITS OF THE PSL ARE REALLY LIKE THOSE HERE
:1043 : EXCEPT FOR THE CONDITION CODES WHICH MUST BE OBTAINED
:1044 : FROM PSL.CC.
:1045 :
:1046 : PSL=1D ; SOFT COPY OF VAX PSL
:1047 : VAR=1E ; VIRTUAL ADDRESS REGISTER
:1048 : VAR2=1F ; VIRTUAL ADDRESS REGISTER
:1049 :
:1050 :
:1051 : #1=20 ; MASK 00000001 (HEX)
:1052 : #2=21 ; MASK 00000002
:1053 : #4=22 ; MASK 00000004
:1054 : #8=23 ; MASK 00000008
:1055 : #10=24 ; MASK 00000010

```

:1056	: #20=25	: MASK 00000020
:1057	: #40=26	: MASK 00000040
:1058	: #80=27	: MASK 00000080
:1059	: #100=28	: MASK 00000100
:1060	: #200=29	: MASK 00000200
:1061	: #400=2A	: MASK 00000400
:1062	: #800=2B	: MASK 00000800
:1063	: #1000=2C	: MASK 00001000
:1064	: #2000=2D	: MASK 00002000
:1065	: #4000=2E	: MASK 00004000
:1066	: #8000=2F	: MASK 00008000
:1067	: #10000=30	: MASK 00010000
:1068	: #20000=31	: MASK 00020000
:1069	: #40000=32	: MASK 00040000
:1070	: #80000=33	: MASK 00080000
:1071	: #100000=34	: MASK 00100000
:1072	: #200000=35	: MASK 00200000
:1073	: #400000=36	: MASK 00400000
:1074	: #800000=37	: MASK 00800000
:1075	: #1000000=38	: MASK 01000000
:1076	: #2000000=39	: MASK 02000000
:1077	: #4000000=3A	: MASK 04000000
:1078	: #8000000=3B	: MASK 08000000
:1079	: #10000000=3C	: MASK 10000000
:1080	: #20000000=3D	: MASK 20000000
:1081	: #40000000=3E	: MASK 40000000
:1082	: #80000000=3F	: MASK 80000000
:1083		
:1084	: BIT0=20	: MASK 00000001
:1085	: BIT1=21	: MASK 00000002
:1086	: BIT2=22	: MASK 00000004
:1087	: BIT3=23	: MASK 00000008
:1088	: BIT4=24	: MASK 00000010
:1089	: BIT5=25	: MASK 00000020
:1090	: BIT6=26	: MASK 00000040
:1091	: BIT7=27	: MASK 00000080
:1092	: BIT8=28	: MASK 00000100
:1093	: BIT9=29	: MASK 00000200
:1094	: BIT10=2A	: MASK 00000400
:1095	: BIT11=2B	: MASK 00000800
:1096	: BIT12=2C	: MASK 00001000
:1097	: BIT13=2D	: MASK 00002000
:1098	: BIT14=2E	: MASK 00004000
:1099	: BIT15=2F	: MASK 00008000
:1100	: BIT16=30	: MASK 00010000
:1101	: BIT17=31	: MASK 00020000
:1102	: BIT18=32	: MASK 00040000
:1103	: BIT19=33	: MASK 00080000
:1104	: BIT20=34	: MASK 00100000
:1105	: BIT21=35	: MASK 00200000
:1106	: BIT22=36	: MASK 00400000
:1107	: BIT23=37	: MASK 00800000
:1108	: BIT24=38	: MASK 01000000
:1109	: BIT25=39	: MASK 02000000
:1110	: BIT26=3A	: MASK 04000000

```

:1111 : BIT27=3B : MASK 08000000
:1112 : BIT28=3C : MASK 10000000
:1113 : BIT29=3D : MASK 20000000
:1114 : BIT30=3E : MASK 40000000
:1115 : BIT31=3F : MASK 80000000
:1116 :
:1117 : GPR0=40 : GPR 0
:1118 : GPR1=41 : GPR 1
:1119 : GPR2=42 : GPR 2
:1120 : GPR3=43 : GPR 3
:1121 : GPR4=44 : GPR 4
:1122 : GPR5=45 : GPR 5
:1123 : GPR6=46 : GPR 6
:1124 : CM.SP=46 : IN COMPATIBILITY MODE THIS IS THE STACK POINTER.
:1125 : GPR7=47 : GPR 7
:1126 : CM.PC=47 : IN COMPATIBILITY MODE THIS IS THE PC.
:1127 : GPR8=48 : GPR 8
:1128 : GPR9=49 : GPR 9
:1129 : GPR10=4A : GPR 10
:1130 : GPR11=4B : GPR 11
:1131 : GPR12=4C : GPR 12
:1132 : AP=4C : ARGUMENT POINTER
:1133 : GPR13=4D : GPR 13
:1134 : FP=4D : FRAME POINTER
:1135 : SP=4E : GPR 14
:1136 : T0=4F : TEMP LOCATION 0
:1137 :
:1138 : ZERO=50 : ZERO CONSTANT
:1139 : #3=51 : CONSTANT
:1140 : #5=52 : CONSTANT
:1141 : #6=53 : CONSTANT
:1142 : #7=54 : CONSTANT
:1143 : #9=55 : CONSTANT
:1144 : #B=56 : CONSTANT
:1145 : #C=57 : CONSTANT
:1146 : #D=58 : CONSTANT
:1147 : #E=59 : CONSTANT
:1148 : #F=5A : CONSTANT
:1149 : #13=5B : CONSTANT
:1150 : #1F=5C : CONSTANT
:1151 : #34=5D : CONSTANT
:1152 : #3B=5E : CONSTANT
:1153 : #3F=5F : CONSTANT
:1154 : #4B=60 : CONSTANT
:1155 : #66=61 : CONSTANT
:1156 : #A0=62 : CONSTANT
:1157 : #F0=63 : CONSTANT
:1158 : #FF=64 : CONSTANT
:1159 : #1FF=65 : CONSTANT
:1160 : #FFF=66 : CONSTANT
:1161 : #2001=67 : CONSTANT
:1162 : #3000=68 : CONSTANT
:1163 : #7F80=69 : CONSTANT
:1164 : #C000=6A : CONSTANT
:1165 : #FFE0=6B : CONSTANT
  
```

```

:1166 : #FFFF=6C : CONSTANT
:1167 : #1F0000=6D : CONSTANT
:1168 : #200001=6E : CONSTANT
:1169 : #F27000=6F : CONSTANT
:1170 : #3000000=70 : CONSTANT
:1171 : #41F0000=71 : CONSTANT
:1172 : #7000000=72 : CONSTANT
:1173 : #3FFFFFF00=73 : CONSTANT
:1174 : #7F800000=74 : CONSTANT
:1175 : #7FFFFFF00=75 : CONSTANT
:1176 : #7FFFFFF0E=76 : CONSTANT
:1177 : #BF800001=77 : CONSTANT
:1178 :
:1179 : GPR.OS=78 : HARDWARE INDEX TO GPRS
:1180 : BIT.OS(3-0)=79 : 16 LOCATION HARDWARE INDEX TO BIT MASKS
:1181 : BIT.OS(4-0)=7B : 32 LOCATION HARDWARE INDEX TO BIT MASKS
:1182 : OS=7C : OS REGISTER
:1183 : DEC.CON=7D : DECIMAL CARRIES
:1184 : SIZE=7E : SIZE REGISTER
:1185 : MDT= 7E : MEMORY REFERENCE DATA SIZE
:1186 : INTERRUPT.VEC=7E : HARDWARE INTERRUPT VECTOR
:1187 :
:1188 : THIS LOCATION IS THE HARDWARE CONDITION CODES. THE CC'S CAN BE
:1189 : READ OR ARE WRITTEN HERE. NO OTHER BITS IN THE PSL ARE EFFECTED.
:1190 :
:1191 : PSL.CC=7F : PSL CONDITION CODES
:1192 :
:1193 : THESE ADDRESSES ARE ONLY ACCESSIBLE WITH THE MOV MICRO INSTRUCTION.
:1194 :
:1195 : KSP=80 : KERNEL STACK POINTER
:1196 : ESP=81 : EXECUTIVE STACK POINTER
:1197 : SSP=82 : SUPERVISOR STACK POINTER
:1198 : USP=83 : USER STACK POINTER
:1199 : ISP=84 : INTERRUPT STACK POINTER
:1200 : ASTLVL=85 : AST LEVEL
:1201 : PCBB=86 : PROCESS CONTROL BLOCK BASE
:1202 : SCBB=87 : SYSTEM CONTROL BLOCK BASE
:1203 : SISR=88 : SOFTWARE INTERRUPT SUMMARY REGISTER
:1204 :
:1205 : CONSOLE.COMMAND=89 : ONE BYTE COMMAND.
:1206 : CONSOLE.MODIFIER=8A : ONE BYTE MODIFIER.
:1207 : CONSOLE.ADDRESS=8B : FOUR BYTES OF ADDRESS.
:1208 : CONSOLE.DATA=8C : FOUR BYTES OF DATA.
:1209 : CONSOLE.STATUS=8D : ONE BYTE OF STATUS.
:1210 : PSEUDO.NICR=8E : COPY OF NICR THAT 8085 HAS.
:1211 : PSEUDO.ICR=8F : COPY OF ICR THAT 8085 HAS.
:1212 : DEBUG.SAVE.WR0=90 : TEMPORARY SAVE WR0.
:1213 : DEBUG.SAVE.WR1=91 : TEMPORARY SAVE WR1.
:1214 : DEBUG.SAVE.ALU.CC=92 : TEMPORARY SAVE CONDITION CODES.
:1215 : MEMORY.SIZE=93 : MEMORY SIZE OF THE MACHINE.
:1216 : CRD.LOG=94 : INFORMATION ABOUT LAST SOFT MEMORY ERROR.
:1217 : PACKET.A=95 : TOP OF PACKET.
:1218 : PACKET.B=96 : BOTTOM OF PACKET.
:1219 : DEBUG.SAVE.ADDRESS=97 : ADDRESS FOR EXAMINE/DEPOSIT.
:1220 :
    
```

:1221 : SAVED.MDT=99 : SAVED ADDRESS OF MDT FOR MMIE OPTIMIZATION.
 :1222 : POBR=9A : P0 BASE REGISTER
 :1223 : POLR=9B : P0 LENGTH REGISTER
 :1224 : P1BR=9C : P1 BASE REGISTER
 :1225 : P1LR=9D : P1 LENGTH REGISTER
 :1226 : SBR=9E : SYSTEM BASE REGISTER
 :1227 : SLR=9F : SYSTEM LENGTH REGISTER

:1228 :
 :1229 : PORT0=0A0 : RESERVED FOR PORT.
 :1230 : PORT1=0A1 : RESERVED FOR PORT.
 :1231 : PORT2=0A2 : RESERVED FOR PORT.
 :1232 : PORT3=0A3 : RESERVED FOR PORT.
 :1233 : PORT4=0A4 : RESERVED FOR PORT.
 :1234 : PORT5=0A5 : RESERVED FOR PORT.
 :1235 : PORT6=0A6 : RESERVED FOR PORT.
 :1236 : PORT7=0A7 : RESERVED FOR PORT.
 :1237 : PORT8=0A8 : RESERVED FOR PORT.
 :1238 : PORT9=0A9 : RESERVED FOR PORT.
 :1239 : PORT10=0AA : RESERVED FOR PORT.
 :1240 : PORT11=0AB : RESERVED FOR PORT.
 :1241 : PORT12=0AC : RESERVED FOR PORT.
 :1242 : PORT13=0AD : RESERVED FOR PORT.
 :1243 : PORT14=0AE : RESERVED FOR PORT.
 :1244 : PORT15=0AF : RESERVED FOR PORT.
 :1245 : PORT16=0B0 : RESERVED FOR PORT.
 :1246 : PORT17=0B1 : RESERVED FOR PORT.
 :1247 : PORT18=0B2 : RESERVED FOR PORT.
 :1248 : PORT19=0B3 : RESERVED FOR PORT.
 :1249 : PORT20=0B4 : RESERVED FOR PORT.
 :1250 : PORT21=0B5 : RESERVED FOR PORT.
 :1251 : PORT22=0B6 : RESERVED FOR PORT.
 :1252 : PORT23=0B7 : RESERVED FOR PORT.
 :1253 : PORT24=0B8 : RESERVED FOR PORT.
 :1254 : PORT25=0B9 : RESERVED FOR PORT.
 :1255 : PORT26=0BA : RESERVED FOR PORT.
 :1256 : PORT27=0BB : RESERVED FOR PORT.
 :1257 : PORT28=0BC : RESERVED FOR PORT.
 :1258 : PORT29=0BD : RESERVED FOR PORT.
 :1259 : PORT30=0BE : RESERVED FOR PORT.
 :1260 : PORT31=0BF : RESERVED FOR PORT.

:1261 :
 :1262 : XGPR0=0C0 : BACKUP COPY OF GPR 0
 :1263 : XGPR1=0C1 : BACKUP COPY OF GPR 1
 :1264 : XGPR2=0C2 : BACKUP COPY OF GPR 2
 :1265 : XGPR3=0C3 : BACKUP COPY OF GPR 3
 :1266 : XGPR4=0C4 : BACKUP COPY OF GPR 4
 :1267 : XGPR5=0C5 : BACKUP COPY OF GPR 5
 :1268 : XGPR6=0C6 : BACKUP COPY OF GPR 6
 :1269 : XGPR7=0C7 : BACKUP COPY OF GPR 7
 :1270 : XGPR8=0C8 : BACKUP COPY OF GPR 8
 :1271 : XGPR9=0C9 : BACKUP COPY OF GPR 9
 :1272 : XGPR10=0CA : BACKUP COPY OF GPR 10
 :1273 : XGPR11=0CB : BACKUP COPY OF GPR 11
 :1274 : XGPR12=0CC : BACKUP COPY OF GPR 12
 :1275 : XGPR13=0CD : BACKUP COPY OF GPR 13

```

:1276 : XSP=0CE ; BACKUP COPY OF GPR 14
:1277 :
:1278 : #14=0D0 ; CONSTANT
:1279 : #18=0D1 ; CONSTANT
:1280 : #1C=0D2 ; CONSTANT
:1281 : #24=0D3 ; CONSTANT
:1282 : #28=0D4 ; CONSTANT
:1283 : #2C=0D5 ; CONSTANT
:1284 : #2D=0D6 ; CONSTANT
:1285 : #30=0D7 ; CONSTANT
:1286 : #F8=0D8 ; CONSTANT
:1287 : #FC=0D9 ; CONSTANT
:1288 : #2D20=0DA ; CONSTANT
:1289 : #140000=0DB ; CONSTANT
:1290 : #150000=0DC ; CONSTANT
:1291 : #160000=0DD ; CONSTANT
:1292 : #170000=0DE ; CONSTANT
:1293 : #180000=0DF ; CONSTANT
:1294 : #1E0000=0E0 ; CONSTANT
:1295 : #40A10=0E1 ; CONSTANT
:1296 : #C0000E0=0E2 ; CONSTANT
:1297 : #0FFF0000=0E3 ; CONSTANT
:1298 : #B020FF00=0E4 ; CONSTANT
:1299 : #FFFFFF10=0E5 ; CONSTANT
:1300 : DEBUG.SAVE.T1=0E6 ; TEMPORARY SAVE T1.
:1301 : DEBUG.SAVE.OS=0E7 ; TEMPORARY SAVE OS.
:1302 : DEBUG.SAVE.SIZE=0E8 ; TEMPORARY SAVE SIZE VALUE.
:1303 : SAVED.WR0=0E9 ; MM SAVED WR0
:1304 : SAVED.WR1=0EA ; MM SAVED WR1
:1305 : SAVED.2ND.VAR=0EB ; MM SAVED VAR 2
:1306 : SAVED.ALU.CC=0EC ; MM SAVED ALU CONDITION CODES
:1307 : SAVED.SIZE=0ED ; MM SAVED LS[SIZE]
:1308 : SAVED.OS=0EE ; MM SAVED LS[OS]
:1309 : SAVED.REQUEST=0EF ; MM SAVED REQUEST DATA
:1310 : SAVED.CRD.LOG=0F0 ; MM SAVED LOCATION
:1311 : OS.BAK=0F1 ; USED BY WARM FLOATING POINT.
:1312 :
:1313 : THIS LOCATION IS FOR USE BY THE MFPR AND MTPR TO THE CONSOLE AND
:1314 : TU58 CSR'S. ALL OF THE INTERRUPT ENABLE BITS ARE HEREIN.
:1315 :
:1316 : CONSOLE/PROGRAM I/O MODE FLAG - BIT<31> 0 - CONSOLE 1 - PROGRAM I/O MODE
:1317 :
:1318 : CRD RETRY IN PROGRESS - BIT<5> 0 - NOTHING DOING 1 - IN PROGRESS
:1319 : MACHINE CHECK IN PROGRESS - BIT<4>
:1320 : TU58 TRANSMIT CSR IE - BIT<3>
:1321 : TU58 RECEIVER CSR IE - BIT<2>
:1322 : CONSOLE TRANSMIT CSR IE - BIT<1>
:1323 : CONSOLE TRANSMIT CSR IE - BIT<0>
:1324 :
:1325 : CONSOLE.CSR.IES=0F2 ; CONSOLE CSR IE'S AND CONSOLE/PROGRAM I/O MODE FLAG.
:1326 : IE.TEMP1=0F3 ; INTERRUPTS AND EXCEPTIONS TEMPORARY LOCATION ONE
:1327 : IE.TEMP2=0F4 ; INTERRUPTS AND EXCEPTIONS TEMPORARY LOCATION TWO
:1328 : IE.TEMP3=0F5 ; INTERRUPTS AND EXCEPTIONS TEMPORARY LOCATION THREE
:1329 :
:1330 : XGPR.OS=0F8 ; HARDWARE INDEX TO XGPRS
  
```



```

:1331 : PORT(3-0)=0F9 ; 16 LOCATION HARDWARE INDEX TO PORT REGISTERS.
:1332 : PORT(4-0)=0FB ; 32 LOCATION HARDWARE INDEX TO PORT REGISTERS.
:1333 : CCR=0FC ; CONSOLE READ REGISTER
:1334 : TIMER=0FD ; INTERVAL TIMER VALUE.
:1335 : ALU.CC=0FE ; ALU CONDITION CODES
:1336 :
:1337 : THIS LOCATION IS A WRITE ONLY REGISTER. THE FOLLOWING BITS IN THE
:1338 : PSL ARE AFFECTED:
:1339 :
:1340 : COMPATIBILITY MODE - BIT<31>
:1341 : CURRENT MODE - BITS<25:24>
:1342 : INTERRUPT PRIORITY LEVEL (IPL) - BITS<20:16>
:1343 : TRACE TRAP ENABLE (REALLY T OR TP) - BIT<4>
:1344 : NEGATIVE CONDITION CODE - BIT<3>
:1345 : ZERO CONDITION CODE - BIT<2>
:1346 : OVERFLOW CONDITION CODE - BIT<1>
:1347 : CARRY CONDITION CODE - BIT<0>
:1348 :
:1349 : PSL.HW=OFF ; HARDWARE PSL BITS
:1350 :
:1351 :
:1352 : DATA PATH CONTROL
:1353 :
:1354 : THIS FIELD IS BUILT FROM AN ADDRESS IN THE CCODE MEMORY. THE CCODE MEMORY
:1355 : CONTAINS DATA PATH MICRO INSTRUCTIONS TO PRODUCE THE DESIRED OPERATION
:1356 : FOR THE UCODE MACROS.
:1357 :
:1358 : DATA PATH CONTROL FIELD FOR THE 'BASIC' MICRO INSTRUCTION
:1359 :
:1360 : DP/= <21:16>, .VALIDITY=<DP.VAL>, .DEFAULT=<.SHIFT[.AND[<SDP/Q.CMP.LS>,OFF],-2]>
:1361 :
:1362 : THIS FIELD IS ONLY USED WITH THE INSTRUCTIONS WHICH CAN HAVE
:1363 : XCHG ATTACHED.
:1364 :
:1365 : DP.SMALL/= <20:16>, .VALIDITY=<DP.VAL>
:1366 :
:1367 : EXCHANGE ORIGINAL WR TO LS
:1368 :
:1369 : XCHG.BIT/= <21>, .DEFAULT=<0>
:1370 :
:1371 : YES=1
:1372 : NO=0
:1373 :
:1374 : DATA PATH CONTROL FIELD FOR THE 'EXTENDED' MICRO INSTRUCTION
:1375 :
:1376 : XDP/= <19:14>, .VALIDITY=<A.VAL>
:1377 :
:1378 : DATA PATH CONTROL FOR THE 'MOVE' MICRO INSTRUCTION
:1379 :
:1380 : MDP/= <19:17>, .VALIDITY=<XD.VAL>
:1381 :

```

```

:1382 .PAGE
:1383 : CONDITION CODE FIELD / DATA PATH CONTEXT
:1384 :
:1385 CC/= <6:5> , .VALIDITY=<CC.VAL> , .DEFAULT=<CC/DT(LONG)&HOLD.ALU.CC>
:1386
:1387 DT(LONG)&HOLD.ALU.CC=0 ; USE LONG CONTEXT(32 BITS) AND HOLD ALU CONDITION CODES
:1388 DT(LONG)&SET.ALU.CC=1 ; USE LONG CONTEXT(32 BITS) AND SET ALU CONDITION CODES
:1389 DT(SIZE)&SET.ALU.CC=2 ; USE CONTEXT IN THE SIZE REGISTER AND SET ALU CONDITION CODES
:1390 DT(SIZE)&MAP.ALU.CC.TO.PSL=3 ; TRANSFER ALU CONDITION CODES TO PSL CONDITION CODES HARDWARE DEPENDENT
:1391
:1392
:1393 : MEMORY DATA TYPE FIELD
:1394 :
:1395 MDT/= <6:5> , .VALIDITY=<DT.VAL>
:1396
:1397 BYTE=0 ; SIZE = BYTE
:1398 WORD=1 ; SIZE = WORD
:1399 SIZE=2 ; SIZE = CONTENTS OF SIZE REGISTER
:1400 LONG=3 ; SIZE = LONG
:1401
:1402
:1403 : SHIFT INPUT FIELD
:1404 :
:1405 : THIS CONTROL FIELD DETERMINES THE SHIFT INPUTS FOR THE FOLLOWING OPERATIONS.
:1406 : THE DIRECTION OF THE SHIFT IS DETERMINED BY THE 2901 DESTINATION CODE.
:1407 :
:1408 SI/= <13:11> , .VALIDITY=<A.VAL>
:1409
:1410 ROT.WR=0 ; ROTATE WR
:1411 ROT.WR.Q=1 ; ROTATE WR AND Q
:1412 ASH.WR=2 ; ARITHMETIC SHIFT OF WR
:1413 ASH.WR.Q=3 ; ARITHMETIC SHIFT OF WR AND Q
:1414 MUL.SHF=4 ; SIGNED MULTIPLY SHIFT OPERATION
:1415 UMUL.SHF=5 ; UNSIGNED MULTIPLY SHIFT OPERATION
:1416 SHF.WR.Q=6 ; LOGICAL SHIFT WR AND Q
:1417
:1418
:1419 : SKIP MICRO CONTROL FIELD
:1420 :
:1421 : THIS FIELD IS VALID FOR ALL INSTRUCTIONS EXCEPT
:1422 :
:1423 : JUMP MICRO INSTRUCTION
:1424 : DECODE MICRO INSTRUCTION
:1425 :
:1426 SCTL/= <4:0> , .VALIDITY=<.AND[<SCTL.VAL>,<SKIP.PAGE.VAL>]> , .DEFAULT=<SCTL/NO.SKIP>
:1427
:1428 N.CLR=0 ; ALU N-BIT EQUAL ZERO
:1429 NEQ=1 ; ALU Z-BIT EQUAL ZERO
:1430 BIT.SET=1 ; ALU Z-BIT EQUAL ZERO
:1431 V.CLR=2 ; ALU V-BIT EQUAL ZERO
:1432 C.SET=3 ; ALU C-BIT EQUAL ONE
:1433 GEQU=3 ; ALU C-BIT EQUAL ONE
:1434 NOT.PSL.C=4 ; PSL C EQUAL ZERO
:1435 BR.FALSE=5 ; BRANCH INSTRUCTION FALSE
:1436 R.DST=6 ; REGISTER MODE FLAG

```

```

:1437 NO.INTERRUPT=7 ; NO INTERRUPT PENDING
:1438 N.SET=8 ; ALU N-BIT EQUAL ONE
:1439 EQL=9 ; ALU Z-BIT EQUAL ONE
:1440 BITS.CLR=9 ; ALU Z-BIT EQUAL ONE
:1441 V.SET=0A ; ALU V-BIT EQUAL ONE
:1442 C.CLR=0B ; ALU C-BIT EQUAL ZERO
:1443 LSSU=0B ; ALU C-BIT EQUAL ZERO
:1444 STATE.ZERO.SET=0C ; STATE REGISTER BIT ZERO TRUE
:1445 STATE.ONE.SET=0D ; STATE REGISTER BIT ONE TRUE
:1446 STATE.ZERO.CLR=0E ; STATE REGISTER BIT ZERO FALSE
:1447 STATE.ONE.CLR=0F ; STATE REGISTER BIT ONE FALSE
:1448 RET.NOERR=10 ; RETURN+1 IF NO MEMORY ERROR
:1449 JMP.Z.CLR=11 ; JUMP TO (STACK) IF ALU Z = 0
:1450 POP.USTACK=12 ; DISCARD TOP STACK ENTRY
:1451 JMP.C.SET=13 ; JUMP TO (STACK) IF ALU C=1
:1452 RETURN=14 ; RETURN TO (USTACK)
:1453 NO.SKIP=15 ; EXECUTE NEXT INSTRUCTION
:1454 RETURN+1=16 ; RETURN TO (USTACK)+1
:1455 LOOP.IF.NO.I.AND.SYNC=17 ; JUMP TO (STACK) IF NO INTERRUPT AND NO ACCELERATOR SYNC.
:1456 CONSOLE.ATTN=18 ; CONSOLE ATTENTION
:1457 CONSOLE.ACK=19 ; CONSOLE ACKNOWLEDGE
:1458 NO.FAST.INTERRUPT=1A ; NO FAST INTERRUPTS PENDING
:1459 BACKUP.MASK.VALID=1B ; REGISTER BACKUP MASK VALID.
:1460 LSS=1C ; SIGNED LESS THAN.
:1461 MEM.REF.OK=1D ; NO MEMORY ERROR TRUE
:1462 SKIP=1E ; EXECUTE ADDRESS PLUS ONE
:1463 SYNC=1F ; ACCELERATOR SYNCHRONIZATION

```

:1464 : JUMP MICRO CONTROL FIELD

:1465 : THIS FIELD IS VALID FOR THE FOLLOWING INSTRUCTIONS

:1466 : JUMP MICRO INSTRUCTION

:1467 : DECODE MICRO INSTRUCTION

:1468 : JCTL/= <3:0>, .VALIDITY=<.AND[<JCTL.VAL>, <JUMP.PAGE.VAL>]>, .DEFAULT=<JCTL/JUMP.VALID>

```

:1474 N.CLR=0 ; ALU N-BIT EQUAL ZERO
:1475 NEQ=1 ; ALU Z-BIT EQUAL ZERO
:1476 BIT.SET=1 ; ALU Z-BIT EQUAL ZERO
:1477 V.CLR=2 ; ALU V-BIT EQUAL ZERO
:1478 C.SET=3 ; ALU C-BIT EQUAL ONE
:1479 GEQU=3 ; ALU C-BIT EQUAL ONE
:1480 JUMP=4 ; JUMP UNCONDITIONALLY
:1481 BR.FALSE=5 ; BRANCH INSTRUCTION FALSE
:1482 R.DST=6 ; REGISTER MODE FLAG
:1483 NO.INTERRUPT=7 ; NO INTERRUPT PENDING
:1484 N.SET=8 ; ALU N-BIT EQUAL ONE
:1485 EQL=9 ; ALU Z-BIT EQUAL ONE
:1486 BITS.CLR=9 ; ALU Z-BIT EQUAL ONE
:1487 V.SET=0A ; ALU V-BIT EQUAL ONE
:1488 C.CLR=0B ; ALU C-BIT EQUAL ZERO
:1489 LSSU=0B ; ALU C-BIT EQUAL ZERO
:1490 JSR=0C ; UNCONDITIONAL JUMP AND PUSH USTACK
:1491

```

```

:1492          JSR.VALID=0D          ; JUMP AND PUSH USTACK IF IB VALID=1
:1493          NO.JUMP.TST=0E        ; DO NOT JUMP AND SKIP IF IB VALID=0
:1494          JUMP.VALID=0F         ; JUMP IF IB VALID=1
:1495
:1496
:1497          ; JUMP ADDRESS FIELD
:1498          ;
:1499          JUMP.ADRS/= <17:4> , .ADDRESS , .VALIDITY = <JUMP.VAL>
:1500
:1501
:1502          ; OS ENABLE
:1503          ;
:1504          OS/= <18> , .VALIDITY = <JUMP.VAL>
:1505
:1506          NOP=0
:1507          WITH.OS=1              ; CONCATENATE OS REGISTER TO JUMP ADDRESS
:1508
:1509

```

```

:1510 .PAGE
:1511 : BACKUP PC
:1512 :
:1513 BPC/= <18> , .VALIDITY=<DECODE.VAL> , .DEFAULT=<.CMSE[<.EQL[<OPC2/> , <OPC2/DECODE>]]>]OF[0,1]>
:1514 :
:1515 NOP=1
:1516 SAVE.PC=0 ; WRITE ALU DATA INTO WR[B] (PC+1)
:1517 :
:1518 :
:1519 : DECODE ADDRESS FIELD
:1520 :
:1521 DEC.ADRS/= <17:12> , .VALIDITY=<DECODE.VAL>
:1522 :
:1523 :
:1524 : IR FUNCTION
:1525 :
:1526 IFUNC/= <11:9> , .VALIDITY=<DECODE.VAL>
:1527 :
:1528 CM.EXEC=0 ; COMPATIBILITY MODE EXECUTION DISPATCH WHEN UPPER BYTE = ZERO
:1529 SPEC=0 ; SPECIFIER DISPATCH
:1530 CM.IRD=1 ; COMPATIBILITY MODE INSTRUCTION DECODE DISPATCH
:1531 FLOAT=1 ; SPECIFIER FLOAT TYPE DISPATCH
:1532 VAX.EXEC=2 ; VAX EXECUTION DISPATCH
:1533 ASRC=2 ; ADDRESS SPECIFIER DISPATCH
:1534 VAX.IRD=3 ; VAX OPCODE DISPATCH
:1535 VSRC=4 ; ADDRESS SPECIFIER DISPATCH FOR BIT FIELD INSTRUCTIONS
:1536 ESRC=5 ; SPECIFIER DISPATCH IN INDEX MODE
:1537 CM.DST=6 ; COMPATIBILITY MODE DESTINATION DISPATCH
:1538 CM.SINGLE=7 ; COMPATIBILITY MODE SOP DISPATCH
:1539 :
:1540 :
:1541 : INSTRUCTION BUFFER REQUEST
:1542 :
:1543 IB.REQ/= <8> , .VALIDITY=<DECODE.VAL> , .DEFAULT=0
:1544 :
:1545 NOP=0
:1546 IB.REQ=1 ; ENABLE HARDWARE DEPENDENT MEMORY REQUEST
:1547 :
:1548 : COMPATIBILITY MODE OPCODE SELECT
:1549 :
:1550 CM.HI/= <7> , .VALIDITY=<DECODE.VAL> , .DEFAULT=0
:1551 :
:1552 LOW.BYTE=0 ; DECODE IR BITS<7:0>
:1553 HI.BYTE=1 ; DECODE IR BITS <15:6>
:1554 :
:1555 :
:1556 : LOAD OS REGISTER
:1557 :
:1558 LD.OS/= <6> , .VALIDITY=<DECODE.VAL> , .DEFAULT=0
:1559 :
:1560 NOP=0
:1561 LOAD.OS=1 ; LOAD OS REGISTER FROM I-BUFFER
:1562 :
:1563 :
:1564 : REGISTER DESTINATION FLAG ENABLE

```

[illegible]

```

:1586 .PAGE
:1587 . MISCELLANEOUS FUNCTIONS
:1588
:1589
:1590 . THIS FIELD IS A FLAG FOR STEERING THE MICROWORD AS A MISCELLANEOUS
:1591 . INSTRUCTION OR AS A PORT INSTRUCTION.
:1592
:1593 MISC.PORT/=<18>
:1594
:1595     MISC=0
:1596     PORT=1
:1597
:1598 MISC.0/=<17>,.VALIDITY=<MISC.VAL>
:1599
:1600     NOP=0 ; NO OPERATION IN ALU OR CONDITION CODES.
:1601
:1602 MISC.1/=<15:12>,.VALIDITY=<MISC.VAL>
:1603
:1604     NOP=0F ; NO OPERATION
:1605     SET.CP.ATTN=0E ; SET CPU ATTENTION FOR CONSOLE
:1606     SET.CP.ACK=0D ; SET CPU ACKNOWLEDGE FOR CONSOLE
:1607     CLR.CP.ATTN.AND.ACK=0C ; CLEAR CPU ATTENTION AND CPU ACKNOWLEDGE
:1608     SET.HIGH.PAGE=0B ; SET BIT FOR HIGH HALF OF WCS.
:1609     CLR.HIGH.PAGE=0A ; CLEAR BIT FOR LOW HALF OF WCS.
:1610     SET.PORT.I.O=9 ; SET PORT I/O MODE
:1611     SET.ACC.I.O=8 ; SET ACCELERATOR I/O MODE
:1612     SET.BACKUP.MASK.VALID=7 ; SET REGISTER BACKUP MASK FLAG
:1613     SET.S1=6 ; SET STATE ONE BIT
:1614     SET.S0=5 ; SET STATE ZERO BIT
:1615     CLR.S1=4 ; CLEAR STATE ONE BIT
:1616     CLR.S0=3 ; CLEAR STATE ZERO BIT
:1617     SET.S1&CLR.S0=2 ; SET STATE ONE BIT AND CLEAR STATE ZERO BIT
:1618     CLR.S1&SET.S0=1 ; CLEAR STATE ONE AND SET STATE ZERO
:1619     CLR=0 ; CLEAR STATE ONE AND STATE ZERO BITS.
:1620
:1621
:1622 MISC.2/=<11:9>,.VALIDITY=<MISC.VAL>
:1623
:1624     NOP=0 ; NO OPERATION
:1625     MASK.INTERRUPTS=1 ; MASK ALL INTERRUPTS (FOR DECODE NEXT CYCLE).
:1626     ASSERT.DA.AND.PASS.WR=2 ; ASSERT DATA AVAILABLE AND PASS WR[0] TO THE ACCELERATOR OR PORT.
:1627     TRAP.ACC=3 ; TRAP ACCELERATOR
:1628     READ.ACC.UPC=4 ; READ ACCELERATOR PROGRAM COUNTER
:1629     TRANSFER.GRANT=5 ; RECOGNIZE PORT REQUEST.
:1630     MASK.HALT.AND.T.BIT=6 ; MASK HALT AND T-BIT TRAPS (FOR DECODE TWO CYCLES LATER)
:1631     WRITE.WR.TO.CONSOLE.REGISTER=7 ; SEND BITS <7:0> TO 8085.
:1632
:1633 MISC.WR/=<8:7>
:1634
:1635 MISC.WRL/=<8>
:1636 MISC.WRR/=<7>
:1637
:1638 PORT.SELECT/=<17:15>
:1639
:1640     IDC=7
  
```


:1641
:1642 PORT.FUNC/= <14:13>
:1643
:1644 READ=0
:1645 WRITE=1
:1646 CONTROL=2
:1647
:1648 R.W.FUNC/= <12:10>
:1649
:1650 CSR=0
:1651 DAR=1
:1652 DATA.BYTE=2
:1653 DATA.WORD=3
:1654 PATTERN=4
:1655 POSITION=5
:1656
:1657 CNTL.FUNC/= <12:10>
:1658
:1659 CLEAR.FIFO.CNTR=0
:1660 RESET.BR=1
:1661 CLEAR.IDC=3
:1662 SET.AUTO=4
:1663 CLEAR.AUTO=5
:1664 SEL.FIFO.A=6
:1665 SEL.FIFO.B=7
:1666

```

:1667 .PAGE
:1668 .MEMORY REQUEST FIELD
:1669
:1670 .THIS FIELD IS USED TO INDICATE THE DESIRED OPERATION TO THE MEMORY CONTROL.
:1671 .NOTE THAT THIS IS A DUMMY FIELD. IT IS ACTUALLY MADE UP OF TWO FIELDS.
:1672 .M.FUNC1 AND M.FUNC2
:1673
:1674 .THE FOLLOWING SPECIAL CONSIDERATIONS MUST BE MADE:
:1675
:1676 .ROTATE.BYTE.RIGHT - PERFORMS 9-BIT ROTATE AND THEREFORE THE
:1677 .ANSWER MUST BE CLEANED UP WITH A ROL WR[ ].
:1678
:1679 .WORD.SWAP - PERFORMS A 15-BIT ROTATE AND THEREFORE
:1680 .THE ANSWER MUST BE CLEANED UP WITH A ROL WR[ ].
:1681
:1682 .OCTA.WRITE.P - MUST BE LONGWORD ALLIGNED.
:1683
:1684 .OCTA.WR.V.WCHK - MUST BE LONGWORD ALLIGNED. THIS DATA CAN
:1685 .CROSS PAGE BOUNDARIES
:1686
:1687 MEM.FUNC/=<7>,.VALIDITY=0
:1688
:1689 .
:1690 .PREFETCH=0 ; USED TO TAKE VAR AND ADD ONE BEFORE REFERENCE.
:1691 .WRITE.TB=01 ; WRITE TRANSLATION BUFFER
:1692 .TEST.V.RCHK=2 ; TEST WITH VIRTUAL ADDRESS AND READ ACCESS CHECK
:1693 .READ.P=3 ; READ WITH PHYSICAL ADDRESS
:1694 .WRITE.P=4 ; WRITE WITH PHYSICAL ADDRESS
:1695 .TEST.V.WCHK=5 ; WRITE WITH VIRTUAL ADDRESS AND WRITE ACCESS CHECK
:1696 .ROTATE.BYTE.RIGHT=6 ; ROTATE ADDRESS DATA 9 BITS RIGHT AND RETURN
:1697 .WORD.SWAP=7 ; ROTATE ADDRESS DATA 15 BITS LEFT AND RETURN
:1698 .READ.V.NOCHK=8 ; READ WITH VIRTUAL ADDRESS AND READ ACCESS CHECK
:1699 .READ.V.WCHK=9 ; READ WITH VIRTUAL ADDRESS AND WRITE ACCESS CHECK
:1700 .READ.V.RCHK=0A ; READ WITH VIRTUAL ADDRESS AND READ ACCESS CHECK
:1701 .READ.MAINT.VAR.INC=0B ; TEST VAR INCREMENTING.
:1702 .WRITE.V.NOCHK=0C ; WRITE WITH VIRTUAL ADDRESS AND NO ACCESS CHECK
:1703 .WRITE.V.WCHK=0D ; WRITE WITH VIRTUAL ADDRESS AND WRITE ACCESS CHECK
:1704 .IB.FILL=0E ; READ WITH VIRTUAL ADDRESS AND READ ACCESS CHECK AND FILL IB
:1705 .READ.V.RCHK.IFILL=0F ; READ WITH VIRTUAL ADDRESS AND READ ACCESS CHECK AND FILL IB
:1706 .WRITE.UBS.MAP=0F ; WRITE TO THE UNIBUS MAP.
:1707
:1708 .OCTA.WRITE.P=10 ; WRITE OCTA DATA WITH PHYSICAL ADDRESS.
:1709 .WRITE.TB.STEP=11 ; WRITE TO TWO TB ENTRIES(WITH CORRECT ADDRESS)
:1710 .READ.UBS.MAP=12 ; READ UNIBUS MAP
:1711 .READ.TB=13 ; READ TRANSLATION BUFFER
:1712 .READ.MAINT.ADRS=14 ; RPUTS VA ON UNIBUS LINES AND READS THIS DATA
:1713 .MAINT.ECC.DATA=15 ; TEST ALL ECC FUNCTIONS.
:1714 .READ.CSR=16 ; READ CONTROL REGISTER
:1715 .READ.CNTRL.REG=16 ; READ CONTROL REGISTER
:1716 .WRITE.CNTRL.REG=17 ; WRITE CONTROL REGISTER
:1717 .WRITE.CSR=17 ; WRITE CONTROL REGISTER
:1718 .OCTA.READ.P=18 ; READ OCTA DATA WITH PHYSICAL ADDRESS.
:1719 .READ.V.WCHK.LOCKED=19 ; READ VIRTUAL ADDRESS AND WRITE ACCESS CHECK AND LOCKED
:1720 .OCTA.READ.V.RCHK=1A ; READ OCTA WITH VIRTUAL ADDRESS AND READ ACCESS CHECK.
:1721 .READ.MAINT.BR.CHECK=1B ; TESTS BRANCHING FUNCTION.
:1722 .ISSUE.BG=1C ; ISSUES BUS GRANT (4+ADDRESS<1:0>)

```

ZZ-ENKCF-1.4	:	ENKCF.MIC	FIELD DEFINITIONS F15-MAR-83	Fiche 1	Frame E4	Sequence 43			ZZ
:	:	ENKCF.MCR	MICRO2 1M(01) 15-MAR-83 11:46:22	ENKCF Micro-diagnostic for IDC module	Rev 01.40	Page	37	:	:
:	:	ENKCF.MIC	FIELD DEFINITIONS FOR MAIN MICRO WORD	VER 00.08				:	:
:1722		OCTA.WR.V.WCHK=1D	; WRITE OCTA WITH VIRTUAL ADDRESS AND WRITE ACCESS CHECK.						:2
:1723		QUAD.READ.V.RCHK=1E	; READ QUAD WITH VIRTUAL ADDRESS AND READ ACCESS CHECK.						:2
:1724		READ.MAINT.UBS=1F	; PUTS VA ON UNIBUS LINES AND READS THIS AS DATA.						:2
:1725									
:1726									
:1727		M.FUNC2/=<18:16>,.VALIDITY=<DT.VAL>							
:1728									
:1729		M.FUNC1/=<8:7>,.VALIDITY=<DT.VAL>							
:1730									
:1731		PAR/=<23>,.DEFAULT=<.PARITY[<OPC2/>,<OS/>,<JUMP.ADRS/>,<JCTL/>]]>							
:1732		.UCODE							

```

:1733 .PAGE 'MACRO DEFINITIONS VER 00.02'
:1734
:1735 THE FORMAT FOR THIS GROUP IS TWO ADDRESS WITH A MODIFY DESTINATION.
:1736 THE FIRST OPERAND IS COMBINED WITH THE SECOND OPERAND AND THE
:1737 RESULT WRITTEN TO THE SECOND OPERAND. CMP AND BIT INSTRUCTIONS ARE
:1738 READ ONLY.
:1739
:1740 DURING ALL ARITHMETIC AND LOGICAL INSTRUCTIONS THE ALU CC'S
:1741 CAN BE LOADED BY ADDING THE
:1742
:1743 DT(SIZE)&SET.ALU.CC
:1744
:1745 MACRO TO THE MICROWORD. THIS SPECIFIES THAT ALU CC'S ARE LOADED
:1746 WITH THE 2901 CONDITIONS. NO EXTERNAL MUNGING IS DONE. IF THE
:1747 DATA OPERATION IS WITH BYTES THEN THE CC'S ARE SET
:1748 ACCORDING TO BITS 7-0. WORD USES BITS 15-0 AND LONGWORD USES BITS
:1749 31-0. THE FOLLOWING IS A DESCRIPTION OF THE ALU CC'S VALUE
:1750 FOR EACH FUNCTION:
:1751
:1752 ADD -- BINARY ADDITION OF THE TWO OPERANDS
:1753
:1754 N - SIGN BIT
:1755 Z - SET IF ANSWER IS ZERO.
:1756 V - ARITHMETIC OVERFLOW
:1757 C - CARRY
:1758
:1759 SUB -- BINARY ADDITION OF FIRST OPERAND AND TWO'S COMPLEMENT OF THE SECOND OPERAND.
:1760
:1761 N - SIGN BIT
:1762 Z - SET IF ANSWER IS ZERO.
:1763 V - ARITHMETIC OVERFLOW
:1764 C - COMPLEMENT OF THE BORROW.
:1765
:1766 BIS -- LOGICAL OR OF THE TWO OPERANDS.
:1767
:1768 N - SIGN BIT
:1769 Z - SET IF ANSWER IS ZERO
:1770 V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1771 C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1772
:1773 BIC -- ONE'S COMPLEMENT OF FIRST OPERAND ANDED WITH SECOND OPERAND.
:1774
:1775 N - SIGN BIT
:1776 Z - SET IF ANSWER IS ZERO
:1777 V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1778 C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1779
:1780 AND -- LOGICAL AND OF THE TWO OPERANDS.
:1781
:1782 N - SIGN BIT
:1783 Z - SET IF ANSWER IS ZERO
:1784 V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1785 C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1786
:1787 XOR -- LOGICAL EXCLUSIVE OR OF THE TWO OPERANDS.

```

```

:1788 :
:1789 :
:1790 : N - SIGN BIT
:1791 : Z - SET IF ANSWER IS ZERO
:1792 : V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1793 : C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1794 :
:1795 : CMP -- PERFORM BINARY ADDITION OF FIRST OPERAND AND TWO'S COMPLEMENT OF SECOND OPERAND BUT DO NOT STORE.
:1796 :
:1797 : N - SIGN BIT
:1798 : Z - SET IF ANSWER IS ZERO.
:1799 : V - ARITHMETIC OVERFLOW
:1800 : C - CARRY
:1801 :
:1802 : BIT -- PERFORM LOGICAL AND OF THE TWO OPERANDS BUT DO NOT STORE.
:1803 :
:1804 : N - SIGN BIT
:1805 : Z - SET IF ANSWER IS ZERO
:1806 : V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1807 : C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1808 :
:1809 : SWAP -- SWITCH THE TWO VALUES.
:1810 :
:1811 : N - SIGN BIT OF VALUE TO WR.
:1812 : Z - SET IF ANSWER IS ZERO OF VALUE TO WR.
:1813 : V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1813 : C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)

```

```

1814 .PAGE
1815 :
1816 : MACROS FOR THE FORM OF WR[B]<-- WR[B] OPERATE LS[D]
1817 :
1818 ADD LS[] TO WR[] 'OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/.SHIFT[<.AND[<SDP/LS.PLUS.WR>,07F]>,-2]>'
1819 SUB LS[] FROM WR[] 'OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/.SHIFT[<.AND[<SDP/LS.FROM.WR>,07F]>,-2]>'
1820 BIS LS[] TO WR[] 'OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/.SHIFT[<.AND[<SDP/LS.OR.WR>,OFF]>,-2]>'
1821 BIC LS[] TO WR[] 'OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/.SHIFT[<.AND[<SDP/LS.MASK.WR>,OFF]>,-2]>'
1822 AND LS[] TO WR[] 'OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/.SHIFT[<.AND[<SDP/LS.AND.WR>,07F]>,-2]>'
1823 XOR LS[] TO WR[] 'OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/.SHIFT[<.AND[<SDP/LS.XOR.WR>,07F]>,-2]>'
1824 :
1825 CMP LS[] WITH WR[] 'OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/.SHIFT[<.AND[<SDP/LS.CMP.WR>,OFF]>,-2]>'
1826 BIT LS[] WITH WR[] 'OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/.SHIFT[<.AND[<SDP/WR.BIT.LS>,OFF]>,-2]>'
1827 SWAP LS[] WITH WR[] 'OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/.SHIFT[<.AND[<SDP/SWAP.LS.WR>,OFF]>,-2]>'
1828 :
1829 : THE FOLLOWING MACROS IS TO BE USED WITH ADD,SUB,
1830 : AND,XOR LS[] TO WR[]
1831 :
1832 : IT WILL TAKE THE ORIGINAL CONTENTS OF WR[]
1833 : AND PUT IT IN LS[].
1834 :
1835 XCHG 'XCHG.BIT/YES'
1836 :
1837 : MACROS FOR THE FORM OF LS[D]<-- LS[D] OPERATE Q-REGISTER
1838 :
1839 ADD Q TO LS[] 'OPC/BASIC,D.ADRS/@1,DP/.SHIFT[<.AND[<SDP/Q.PLUS.LS>,OFF]>,-2]>'
1840 SUB Q FROM LS[] 'OPC/BASIC,D.ADRS/@1,DP/.SHIFT[<.AND[<SDP/Q.FROM.LS>,OFF]>,-2]>'
1841 BIS Q TO LS[] 'OPC/BASIC,D.ADRS/@1,DP/.SHIFT[<.AND[<SDP/Q.OR.LS>,OFF]>,-2]>'
1842 AND Q TO LS[] 'OPC/BASIC,D.ADRS/@1,DP/.SHIFT[<.AND[<SDP/Q.AND.LS>,OFF]>,-2]>'
1843 XOR Q TO LS[] 'OPC/BASIC,D.ADRS/@1,DP/.SHIFT[<.AND[<SDP/Q.XOR.LS>,OFF]>,-2]>'
1844 :
1845 CMP Q WITH LS[] 'OPC/BASIC,D.ADRS/@1,DP/.SHIFT[<.AND[<SDP/Q.CMP.LS>,OFF]>,-2]>'
1846 BIT Q WITH LS[] 'OPC/BASIC,D.ADRS/@1,DP/.SHIFT[<.AND[<SDP/LS.BIT.Q>,OFF]>,-2]>'
1847 :
1848 : MACROS FOR THE FORM OF Q-REGISTER<-- Q-REGISTER OPERATE LS[D]
1849 :
1850 ADD LS[] TO Q 'OPC/BASIC,D.ADRS/@1,DP/.SHIFT[<.AND[<SDP/LS.PLUS.Q>,OFF]>,-2]>'
1851 SUB LS[] FROM Q 'OPC/BASIC,D.ADRS/@1,DP/.SHIFT[<.AND[<SDP/LS.FROM.Q>,OFF]>,-2]>'
1852 BIS LS[] TO Q 'OPC/BASIC,D.ADRS/@1,DP/.SHIFT[<.AND[<SDP/LS.OR.Q>,OFF]>,-2]>'
1853 BIC LS[] TO Q 'OPC/BASIC,D.ADRS/@1,DP/.SHIFT[<.AND[<SDP/LS.MASK.Q>,OFF]>,-2]>'
1854 AND LS[] TO Q 'OPC/BASIC,D.ADRS/@1,DP/.SHIFT[<.AND[<SDP/LS.AND.Q>,OFF]>,-2]>'
1855 XOR LS[] TO Q 'OPC/BASIC,D.ADRS/@1,DP/.SHIFT[<.AND[<SDP/LS.XOR.Q>,OFF]>,-2]>'
1856 :
1857 CMP LS[] WITH Q 'OPC/BASIC,D.ADRS/@1,DP/.SHIFT[<.AND[<SDP/LS.CMP.Q>,OFF]>,-2]>'
1858 BIT LS[] WITH Q 'OPC/BASIC,D.ADRS/@1,DP/.SHIFT[<.AND[<SDP/LS.BIT.Q>,OFF]>,-2]>'
1859 :
1860 : MACROS FOR THE FORM OF LS[D]<-- LS[D] OPERATE WR[B]
1861 :
1862 ADD WR[] TO LS[] 'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/.SHIFT[<.AND[<SDP/WR.PLUS.LS>,OFF]>,-2]>'
1863 SUB WR[] FROM LS[] 'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/.SHIFT[<.AND[<SDP/WR.FROM.LS>,OFF]>,-2]>'
1864 BIS WR[] TO LS[] 'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/.SHIFT[<.AND[<SDP/WR.OR.LS>,OFF]>,-2]>'
1865 AND WR[] TO LS[] 'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/.SHIFT[<.AND[<SDP/WR.AND.LS>,OFF]>,-2]>'
1866 XOR WR[] TO LS[] 'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/.SHIFT[<.AND[<SDP/WR.XOR.LS>,OFF]>,-2]>'
1867 :
1868 CMP WR[] WITH LS[] 'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/.SHIFT[<.AND[<SDP/WR.CMP.LS>,OFF]>,-2]>'
    
```

```

:1869 BIT WR[] WITH LS[]      'OPC1/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.BIT.LS>,OFF]>,-2]>'
:1870 SWAP WR[] WITH LS[]    'SWAP LS[@2] WITH WR[@1]'
:1871 :
:1872 :      MACROS FOR THE FORM OF WR[B]<-- WR[B] OPERATE WR[A]
:1873 :
:1874 ADD WR[] TO WR[]        'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.PLUS.WR>,03F]>'
:1875 SUB WR[] FROM WR[]     'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.FROM.WR>,03F]>'
:1876 BIS WR[] TO WR[]       'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.OR.WR>,03F]>'
:1877 BIC WR[] TO WR[]       'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.MASK.WR>,03F]>'
:1878 AND WR[] TO WR[]       'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.AND.WR>,03F]>'
:1879 XOR WR[] TO WR[]       'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.XOR.WR>,03F]>'
:1880 :
:1881 CMP WR[] WITH WR[]     'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.CMP.WR>,03F]>'
:1882 BIT WR[] WITH WR[]    'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.BIT.WR>,03F]>'
:1883 :
:1884 :      MACROS FOR THE FORM OF WR[B]<-- WR[B] OPERATE Q-REGISTER
:1885 :
:1886 ADD Q TO WR[]          'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/Q.PLUS.WR>,03F]>'
:1887 SUB Q FROM WR[]       'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/Q.FROM.WR>,03F]>'
:1888 BIS Q TO WR[]         'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/Q.OR.WR>,03F]>'
:1889 AND Q TO WR[]        'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/Q.AND.WR>,03F]>'
:1890 XOR Q TO WR[]        'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/Q.XOR.WR>,03F]>'
:1891 :
:1892 CMP Q WITH WR[]       'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/Q.CMP.WR>,03F]>'
:1893 BIT Q WITH WR[]      'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/Q.BIT.WR>,03F]>'
:1894 :
:1895 :      MACROS FOR THE FORM OF Q-REGISTER<-- Q-REGISTER OPERATE WR[B]
:1896 :
:1897 ADD WR[] TO Q          'OPC1/EXTENDED,A.ADRS/@1,XDP/<.AND[<SDP/WR.PLUS.Q>,03F]>'
:1898 SUB WR[] FROM Q       'OPC1/EXTENDED,A.ADRS/@1,XDP/<.AND[<SDP/WR.FROM.Q>,03F]>'
:1899 BIS WR[] TO Q        'OPC1/EXTENDED,A.ADRS/@1,XDP/<.AND[<SDP/WR.OR.Q>,03F]>'
:1900 BIC WR[] TO Q        'OPC1/EXTENDED,A.ADRS/@1,XDP/<.AND[<SDP/WR.MASK.Q>,03F]>'
:1901 AND WR[] TO Q        'OPC1/EXTENDED,A.ADRS/@1,XDP/<.AND[<SDP/WR.AND.Q>,03F]>'
:1902 XOR WR[] TO Q       'OPC1/EXTENDED,A.ADRS/@1,XDP/<.AND[<SDP/WR.XOR.Q>,03F]>'
:1903 :
:1904 CMP WR[] WITH Q      'OPC1/EXTENDED,A.ADRS/@1,XDP/<.AND[<SDP/WR.CMP.Q>,03F]>'
:1905 BIT WR[] WITH Q     'OPC1/EXTENDED,A.ADRS/@1,XDP/<.AND[<SDP/Q.BIT.WR>,03F]>'
  
```



```

:1906 .PAGE
:1907
:1908 THE FORMAT OF THIS GROUP IS THREE ADDRESS TYPE. THE FIRST AND SECOND
:1909 OPERANDS ARE READ AND THE RESULT IS WRITTEN INTO THE THIRD OPERAND.
:1910
:1911 MACROS FOR THE FORM OF Q-REGISTER<-- WR[B] OPERATE WR[A]
:1912
:1913 ADD WR[] PLUS WR[] TO Q 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.PLUS.WR.Q>,03F]>'
:1914 SUB WR[] FROM WR[] TO Q 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.FROM.WR.Q>,03F]>'
:1915 BIS WR[] WITH WR[] TO Q 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.OR.WR.Q>,03F]>'
:1916 BIC WR[] WITH WR[] TO Q 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.MASK.WR.Q>,03F]>'
:1917 AND WR[] WITH WR[] TO Q 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.AND.WR.Q>,03F]>'
:1918 XOR WR[] WITH WR[] TO Q 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.XOR.WR.Q>,03F]>'
:1919
:1920 MACROS FOR THE FORM OF Q-REGISTER<-- WR[B] OPERATE LS[D]
:1921
:1922 ADD WR[] PLUS LS[] TO Q 'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.PLUS.LS.Q>,OFF]>,-2]>'
:1923 SUB WR[] FROM LS[] TO Q 'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.FROM.LS.Q>,OFF]>,-2]>'
:1924 BIS WR[] WITH LS[] TO Q 'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.OR.LS.Q>,OFF]>,-2]>'
:1925 AND WR[] WITH LS[] TO Q 'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.AND.LS.Q>,OFF]>,-2]>'
:1926 XOR WR[] WITH LS[] TO Q 'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.XOR.LS.Q>,OFF]>,-2]>'
:1927
:1928 MACROS FOR THE FORM OF Q-REGISTER <-- LS[D] OPERATE WR[B]
:1929
:1930 ADD LS[] PLUS WR[] TO Q 'ADD WR[@2] PLUS LS[@1] TO Q'
:1931 SUB LS[] FROM WR[] TO Q 'OPC/BASIC,B.ADRS/@2,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/LS.FROM.WR.Q>,OFF]>,-2]>'
:1932 BIS LS[] WITH WR[] TO Q 'BIS WR[@2] WITH LS[@1] TO Q'
:1933 AND LS[] WITH WR[] TO Q 'AND WR[@2] WITH LS[@1] TO Q'
:1934 XOR LS[] WITH WR[] TO Q 'XOR WR[@2] WITH LS[@1] TO Q'
:1935
:1936
    
```

```

:1937 .PAGE
:1938 ; SPECIAL FUNCTION ADD/SUB OPERATIONS
:1939
:1940 ADD (WR[] TO WR[])+1 'DPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.PLUS.WR+1>,03F]>'
:1941 ADD (LS[] TO WR[])+1 'DPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/LS.PLUS.WR+1>,OFF]>,-2]>'
:1942 ADD (WR[] TO LS[])+1 'DPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.PLUS.LS+ OFF]>,-2]>'
:1943
:1944 ADD OS PLUS WR[] TO LS[] 'DPC1/MOVE,B.ADRS/@1,XD.ADRS/@2,MDP/<.SHIFT[<.AND[<SDP/WR.PLUS.OS>,3F]>,-3]>'
:1945
:1946 SUB (WR[] FROM WR[])-1 'DPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.FROM.WR-1>,03F]>'
:1947 SUB (LS[] FROM WR[])-1 'DPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/LS.FROM.WR-1>,OFF]>,-2]>'
:1948 SUB (WR[] FROM LS[])-1 'DPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.FROM.LS-1>,OFF]>,-2]>'
:1949
:1950
:1951 SUB WRB[] FROM WRA[] TO WRB[] 'DPC1/EXTENDED,B.ADRS/@1,A.ADRS/@2,XDP/<.AND[<SDP/WR.FROM.WR.WR>,03F]>'
:1952 SUB LS[] FROM WR[] TO LS 'DPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/LS.FROM.WR.LS>,OFF]>,-2]>'
:1953 SUB WR[] FROM LS[] TO WR 'DPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WRA.FROM.LS.WRB>,OFF]>,-2]>'
:1954
:1955 SUB WRA[] FROM Q TO WRB[] 'DPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XD?/<.AND[<SDP/WR.FROM.Q.WR>,03F]>'
:1956 SUB LS[] FROM Q TO LS[] 'DPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/LS.FROM.Q.LS>,OFF]>,-2]>'
:1957 SUB Q FROM WR[] TO Q 'DPC1/EXTENDED,B.ADRS/@1,XDP/<.AND[<SDP/Q.FROM.WR.Q>,03F]>'
:1958 SUB Q FROM LS[] TO Q 'DPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/Q.FROM.LS.Q>,OFF]>,-2]>'
:1959
:1960 ADD Q PLUS WR[] TO LS[] 'DPC/BASIC,D.ADRS/@2,B.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/Q.PLUS.WR.TO.LS>,OFF]>,-2]>'
:1961 SUB Q FROM WR[] TO LS[] 'DPC/BASIC,D.ADRS/@2,B.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/Q.FROM.WR.TO.LS>,OFF]>,-2]>'
:1962 ADD Q PLUS LS[] TO WR[] 'DPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/Q.PLUS.LS.TO.WR>,OFF]>,-2]>'
:1963
:1964 ADD Q TO WR[] XCHG TO LS[] 'DPC/BASIC,D.ADRS/@2,B.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/WR.PLUS.Q.XCHG>,OFF]>,-2]>'
:1965
  
```

```

:1966 .PAGE
:1967 .MOVE MACRO INSTRUCTIONS
:1968
:1969 .DURING THE MOVE INSTRUCTIONS THE ALU CC'S CAN BE LOADED BY
:1970 .ADDING THE
:1971
:1972 .DT(SIZE)&SET.ALU.CC
:1973
:1974 .MACRO TO THE MICROWORD. THE FOLLOWING IS A DESCRIPTION OF THE ALU
:1975 .CC'S VALUE FOR EACH FUNCTION:
:1976
:1977 .MOV -- PUT FIRST OPERAND ON TOP OF THE SECOND OPERAND.
:1978
:1979 .N - SIGN BIT
:1980 .Z - SET IF ANSWER IS ZERO
:1981 .V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1982 .C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1983
:1984 .MCOM -- PUT ONE'S COMPLEMENT OF THE FIRST OPERAND ON TOP OF THE SECOND OPERAND.
:1985
:1986 .N - SIGN BIT
:1987 .Z - SET IF ANSWER IS ZERO
:1988 .V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1989 .C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1990
:1991 .MNEG -- PUT TWO'S COMPLEMENT OF THE FIRST OPERAND ON TOP OF THE SECOND OPERAND.
:1992
:1993 .N - SIGN BIT
:1994 .Z - SET IF ANSWER IS ZERO.
:1995 .V - ARITHMETIC OVERFLOW
:1996 .C - CARRY
:1997 .MOV Q TO LS[] 'OPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/MOV.Q.LS>,OFF]>,-2]>'
:1998 .MOV Q TO WR[] 'OPC1/EXTENDED,B.ADRS/@1,XDP/<.AND[<SDP/MOV.Q.WR>,03F]>'
:1999 .MOV LS[] TO Q 'OPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/MOV.LS.Q>,OFF]>,-2]>'
:2000 .MOV Q TO WR[] XCHG TO LS[] 'OPC/BASIC,D.ADRS/@2,B.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/MOV.Q.WR&WR.LS>,OFF]>,-2]>'
:2001
:2002 .MOV IB.DATA TO OS 'OPC2/DECODE,IFUNC/SPEC,R.DST/NOP,IB.REQ/IB.REQ,JCTL/NO.JUMP.TST,LD.OS/LOAD.OS'
:2003 .MOV CM.IB.DATA TO OS 'OPC2/DECODE,IFUNC/SPEC,R.DST/NOP,IB.REQ/NOP,JCTL/NO.JUMP.TST,LD.OS/LOAD.OS'
:2004
:2005 .MOV MEM.DATA TO WR[] 'OPC1/MOVE,B.ADRS/@1,MDP/<.SHIFT[<.AND[<SDP/MOV.MEM.WR>,3F]>,-3]>,XD.ADRS/WR.BACKUP'
:2006 .MOV MEM.DATA TO WR[] XCHG TO LS[] 'OPC1/MOVE,B.ADRS/@1,MDP/<.SHIFT[<.AND[<SDP/MOV.MEM.WR>,3F]>,-3]>,XD.ADRS/@2'
:2007 .MOV MEM.DATA TO LS[] 'OPC1/MOVE,XD.ADRS/@1,MDP/<.SHIFT[<.AND[<SDP/MOV.MEM.LS>,3F]>,-3]>'
:2008 .MOV LS[] TO WR[] 'OPC1/MOVE,XD.ADRS/@1,B.ADRS/@2,MDP/<.SHIFT[<.AND[<SDP/MOV.LS.WR>,3F]>,-3]>'
:2009 .MOV WR[] TO LS[] 'OPC1/MOVE,B.ADRS/@1,XD.ADRS/@2,MDP/<.SHIFT[<.AND[<SDP/MOV.WR.LS>,3F]>,-3]>'
:2010 .MOV WR[] TO WR[] 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.PLUS.0.TO.WR>,03F]>'
:2011 .MOV WR[] TO Q 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/WR.OR.WR.Q>,03F]>'
:2012 .MOV XWR[] TO Q 'OPC1/MOVE,XB.ADRS/@1,XD.ADRS/0,MDP/<.SHIFT[<.AND[<SDP/MOV.XWR.Q>,3F]>,-3]>'
:2013
:2014 .MOV FPA TO LS[] 'OPC1/MOVE,XD.ADRS/@1,MDP/<.SHIFT[<.AND[<SDP/MOV.ACC.LS>,3F]>,-3]>'
:2015 .MOV PORT TO LS[] 'OPC1/MOVE,XD.ADRS/@1,MDP/<.SHIFT[<.AND[<SDP/MOV.ACC.LS>,3F]>,-3]>,CC/DT(LONG)&HOLD.ALU.CC'
:2016
:2017 .MCOM WR[] TO LS[] 'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.COM.LS>,OFF]>,-2]>'
:2018 .MCOM LS[] TO WR[] 'OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/LS.COM.WR>,OFF]>,-2]>'
:2019 .MNEG WR[] TO LS[] 'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.NEG.LS>,OFF]>,-2]>'
:2020 .MNEG LS[] TO WR[] 'OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/LS.NEG.WR>,OFF]>,-2]>'

```

C C

[illegible]

```

:2081 : C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:2082 :
:2083 : TST -- ADD ZERO TO THE OPERAND TO SET THE CONDITION CODES.
:2084 :
:2085 : N - SIGN BIT
:2086 : Z - SET IF ANSWER IS ZERO
:2087 : V - ARITHMETIC OVERFLOW (IN THIS CASE ALWAYS ZERO)
:2088 : C - CARRY (IN THIS CASE ZERO)
:2089 :
:2090 :
:2091 CLR Q 'OPC1/EXTENDED,A.ADRS/0,B.ADRS/0,XDP/<.AND[<SDP/WR.XOR.WR.Q>,03F]>'
:2092 CLR LS[] 'OPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/CLR.LS>,OFF]>,-2]>'
:2093 CLR WR[] 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/WR.XOR.WR>,03F]>'
:2094 :
:2095 NEG Q 'OPC1/EXTENDED,XDP/<.AND[<SDP/NEG.Q>,03F]>'
:2096 NEG LS[] 'OPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/NEG.LS>,OFF]>,-2]>'
:2097 NEG WR[] 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/WR.NEG.WR>,03F]>'
:2098 :
:2099 INC Q 'OPC1/EXTENDED,XDP/<.AND[<SDP/INC.Q>,03F]>'
:2100 INC LS[] 'OPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/INC.LS>,OFF]>,-2]>'
:2101 INC WR[] 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/WR.INC.WR>,03F]>'
:2102 :
:2103 DEC Q 'OPC1/EXTENDED,XDP/<.AND[<SDP/DEC.Q>,03F]>'
:2104 DEC LS[] 'OPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/DEC.LS>,OFF]>,-2]>'
:2105 DEC WR[] 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/WR.DEC.WR>,03F]>'
:2106 :
:2107 COM Q 'OPC1/EXTENDED,XDP/<.AND[<SDP/COM.Q>,03F]>'
:2108 COM LS[] 'OPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/COM.LS>,OFF]>,-2]>'
:2109 COM WR[] 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/WR.COM.WR>,03F]>'
:2110 :
:2111 TST WR[] 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/WR.PLUS.0.TO.WR>,03F]>'
:2112 TST Q 'OPC1/EXTENDED,XDP/<.AND[<SDP/Q.PLUS.0>,03F]>'
:2113 :
:2114 NOP 'OPC/BASIC,D.ADRS/0,B.ADRS/0,DP/<.SHIFT[<.AND[<SDP/Q.CMP.LS>,OFF]>,-2]>'
  
```

```

:2115 .PAGE
:2116
:2117 MACROS FOR SHIFT OPERATION ON WR[B] AND/OR Q-REGISTER
:2118
:2119 DURING THE SHIFT OPERATIONS THE ALU CC'S CAN BE LOADED BY
:2120 USING THE
:2121
:2122 DT(SIZE)&SET.ALU.CC
:2123
:2124 MACRO WITH THE MICROWORD. THE FOLLOWING IS A DESCRIPTION OF THE
:2125 ALU CC'S WITH EACH FUNCTION.
:2126
:2127 ROR WR[] -- ROTATE 32 BIT VALUE ONE POSITION RIGHT.
:2128
:2129 N - SIGN OF INPUT
:2130 Z - SET IF INPUT IS ZERO
:2131 V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:2132 C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:2133
:2134 ROL WR[] -- ROTATE 32 BIT VALUE ONE POSITION LEFT.
:2135
:2136 N - SIGN OF INPUT
:2137 Z - SET IF INPUT IS ZERO
:2138 V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:2139 C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:2140
:2141 ASHR WR[] -- SHIFT 32 BIT VALUE RIGHT ONE POSITION AND SIGN EXTEND.
:2142
:2143 N - SIGN OF INPUT
:2144 Z - SET IF INPUT IS ZERO
:2145 V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:2146 C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:2147
:2148 ASHL WR[] -- SHIFT 32 BIT VALUE LEFT ONE POSITION AND INSERT A ZERO IN BOTTOM BIT.
:2149
:2150 N - SIGN OF INPUT
:2151 Z - SET IF INPUT IS ZERO
:2152 V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:2153 C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:2154
:2155 ASHRC WR[].Q -- SHIFT 64 BIT WR.Q ONE POSITION LEFT AND SIGN EXTEND.
:2156
:2157 N - SIGN OF INPUT WR[]
:2158 Z - SET IF INPUT WR[] IS ZERO
:2159 V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:2160 C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:2161
:2162 RORC WR[].Q -- ROTATE 64 BIT WR.Q ONE POSITION RIGHT.
:2163
:2164 N - SIGN OF INPUT WR[]
:2165 Z - SET IF INPUT WR[] IS ZERO
:2166 V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:2167 C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:2168
:2169 ASHLC WR[].Q -- SHIFT 64 BIT WR.Q ONE POSITION LEFT AND INSERT ZERO.

```

```

:2170 :
:2171 : N - SIGN OF INPUT WR[]
:2172 : Z - SET IF INPUT WR[] IS ZERO
:2173 : V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:2174 : C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:2175 :
:2176 : ROLC WR[].Q -- ROTATE 64 BIT WR.Q ONE POSITION LEFT.
:2177 :
:2178 : N - SIGN OF INPUT WR[]
:2179 : Z - SET IF INPUT WR[] IS ZERO
:2180 : V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:2181 : C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:2182 :
:2183 : SHL2 WR[] -- SHIFT 32 BIT VALUE TWO POSITIONS LEFT INSERTING ZEROES IN THE BOTTOM BITS.
:2184 :
:2185 : N - SIGN OF INPUT WR[]+WR[]
:2186 : Z - SET IF INPUT WR[]+WR[] IS ZERO
:2187 : V - OVERFLOW OF INPUT WR[]+WR[]
:2188 : C - CARRY OF INPUT WR[]+WR[]
:2189 :
:2190 ROR WR[] 'OPC1/EXTENDED,B.ADRS/@1,SI/ROT.WR,XDP/<.AND[<SDP/ASHR>,03F]>'
:2191
:2192 ROL WR[] 'OPC1/EXTENDED,B.ADRS/@1,SI/ROT.WR,XDP/<.AND[<SDP/ASHL>,03F]>'
:2193
:2194
:2195 ASHR WR[] 'OPC1/EXTENDED,B.ADRS/@1,SI/ASH.WR,XDP/<.AND[<SDP/ASHR>,03F]>'
:2196
:2197 ASHL WR[] 'OPC1/EXTENDED,B.ADRS/@1,SI/ASH.WR,XDP/<.AND[<SDP/ASHL>,03F]>'
:2198
:2199 ASHRC WR[].Q 'OPC1/EXTENDED,B.ADRS/@1,SI/ASH.WR.Q,XDP/<.AND[<SDP/ASHRC>,03F]>'
:2200
:2201 RORC WR[].Q 'OPC1/EXTENDED,B.ADRS/@1,SI/ROT.WR.Q,XDP/<.AND[<SDP/ASHRC>,03F]>'
:2202
:2203 ASHLC WR[].Q 'OPC1/EXTENDED,B.ADRS/@1,SI/ASH.WR.Q,XDP/<.AND[<SDP/ASHLC>,03F]>'
:2204
:2205 ROLC WR[].Q 'OPC1/EXTENDED,B.ADRS/@1,SI/ROT.WR.Q,XDP/<.AND[<SDP/ASHLC>,03F]>'
:2206
:2207 SHL2 WR[] 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,SI/2,XDP/<.AND[<SDP/SHL2>,03F]>'
:2208
:2209 COND.ADD&SHIFT WR[] TO WR[].Q 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,SI/MUL.SHF,XDP/<.AND[<SDP/COND.ADD&SHIFT>,03F]>'
:2210
:2211 COND.SUB&SHIFT WR[] FROM WR[].Q 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,SI/MUL.SHF,XDP/<.AND[<SDP/COND.SUB&SHIFT>,03F]>'
:2212
:2213 UNSIGNED.MUL WR[] TO WR[].Q 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,SI/UMUL.SHF,XDP/<.AND[<SDP/COND.ADD&SHIFT>,03F]>'
:2214 SHR WR[] 'OPC1/EXTENDED,B.ADRS/@1,SI/SHF.WR.Q,XDP/<.AND[<SDP/ASHR>,03F]>'
:2215 SHL WR[] 'ASHL WR[@1]'
:2216 SHRC WR[].Q 'OPC1/EXTENDED,B.ADRS/@1,SI/SHF.WR.Q,XDP/<.AND[<SDP/ASHRC>,03F]>'
:2217 SHLC WR[].Q 'ASHLC WR[].Q'
:2218
:2219
:2220 ; MEMORY REQUEST MACRO
:2221
:2222 ; * THIS MACRO IS NOT FOR GENERAL USE!!! *
:2223 ; * THIS IS ONLY FOR USE IN THE FOLLOWING MACRO *
:2224 MEM.FUNC[] 'M.FUNC2/<.SHIFT[<MEM.FUNC/@1>,-2]>,M.FUNC1/<.AND[<MEM.FUNC/@1>,3]>'

```



```
:2225  
:2226 MEM.REQ[] ADRS[] DT[] 'OPC2/MEM.REQ,MEM.FUNC[@1],D.ADRS/@2,MDT/@3'  
:2227 WRITE.MEM LS[] 'OPC1/MOVE,XD.ADRS/@1,MDP/<.SHIFT[<.AND[<SDP/MOV.LS.MEM>,3F]>,-3]>'
```

```

:2228 .PAGE
:2229
:2230 : MACROS FOR MISCELLANEOUS OPERATIONS
:2231
:2232 : IN THE NEBULA MICROPROGRAMMING LANGUAGE THERE IS A NEED FOR
:2233 : VARIOUS UNIQUE FUNCTIONAL OPERATIONS. THESE ARE FOUND IN THE
:2234 : MISC INSTRUCTIONS. MOST FUNCTIONS ARE EXECUTED BY INSERTING
:2235 : THE REQUEST INSIDE THE MISC[] AND MISC2[] MACROS.
:2236
:2237 MISC [] 'OPC2/MISC,MISC.1/@1,MISC.2/NOP,MISC.PORT/MISC''
:2238 MISC2 [] 'OPC2/MISC,MISC.1/NOP,MISC.2/@1,MISC.PORT/MISC''
:2239 MISC [] WR[] 'MISC [@1],MISC.WRL/0,MISC.WRR/@2''
:2240 MISC2 [] WR[] 'MISC2 [@1],MISC.WRL/0,MISC.WRR/@2''
:2241
:2242 PORT.CONTROL [] 'OPC2/MISC,MISC.PORT/PORT,CNTL.FUNC/@1,PORT.SELECT/IDC,PORT.FUNC/CONTROL''
:2243 PORT.WRITE PORT.ADRS[] WITH WR[] 'OPC2/MISC,MISC.PORT/PORT,R.W.FUNC/@1,PORT.SELECT/IDC,PORT.FUNC/WRITE,MISC.WRL/0,MIS
:2244 PORT.READ PORT.ADRS[] 'OPC2/MISC,MISC.PORT/PORT,R.W.FUNC/@1,PORT.SELECT/IDC,PORT.FUNC/READ''
:2245
:2246 : A FEW CASES OF THE MISCELLANEOUS FUNCTIONS HAVE BEEN CALLED OUT AS
:2247 : UNIQUE FUNCTIONS IN THE MICROPROGRAMMING LANGUAGE SET. THESE
:2248 : ARE LISTED BELOW.
:2249
:2250 : THE FOLLOWING TO MACROS SEND DATA TO EXTERNAL ACCELERATORS.
:2251
:2252 ASSERT.DA.AND.PASS.WRO 'MISC2 [ASSERT.DA.AND.PASS.WR] WR[0]''
:2253 ASSERT.DA.AND.PASS.WR1 'MISC2 [ASSERT.DA.AND.PASS.WR] WR[1]''
:2254
:2255 : STATE REGISTER MACROS - THE POSSIBLE CHANGES TO THE STATE
:2256 : BITS ARE SPECIFIED AS INDIVIDUAL FUNCTIONS.
:2257
:2258 CLR.STATE.ZERO 'OPC2/MISC,MISC.1/CLR.S0,MISC.2/NOP,MISC.PORT/MISC''
:2259 CLR.STATE.ONE 'OPC2/MISC,MISC.1/CLR.S1,MISC.2/NOP,MISC.PORT/MISC''
:2260 SET.STATE.ZERO 'OPC2/MISC,MISC.1/SET.S0,MISC.2/NOP,MISC.PORT/MISC''
:2261 SET.STATE.ONE 'OPC2/MISC,MISC.1/SET.S1,MISC.2/NOP,MISC.PORT/MISC''
:2262 SET.STATE.ZERO&CLR.STATE.ONE 'OPC2/MISC,MISC.1/CLR.S1&SET.S0,MISC.2/NOP,MISC.PORT/MISC''
:2263 SET.STATE.ONE&CLR.STATE.ZERO 'OPC2/MISC,MISC.1/SET.S1&CLR.S0,MISC.2/NOP,MISC.PORT/MISC''
:2264 CLR.STATE.ONE&CLR.STATE.ZERO 'OPC2/MISC,MISC.1/CLR,MISC.2/NOP,MISC.PORT/MISC''
:2265 SET.BACKUP.MASK.VALID 'OPC2/MISC,MISC.1/SET.BACKUP.MASK.VALID,MISC.2/NOP,MISC.PORT/MISC''
:2266
:2267
:2268 ; CONTEXT SIZE AND CONDITION CODE MACROS
:2269
:2270 DT(SIZE)&MAP.ALU.CC.TO.PSL 'CC/DT(SIZE)&MAP.ALU.CC.TO.PSL''
:2271 DT(LONG)&SET.ALU.CC 'CC/DT(LONG)&SET.ALU.CC''
:2272 DT(SIZE)&SET.ALU.CC 'CC/DT(SIZE)&SET.ALU.CC''
:2273

```

```

:2274 .PAGE
:2275 .JUMP AND MICRO SEQUENCER CONTROL MACROS
:2276
:2277 THE FOLLOWING MACROS ARE USED TO CONTROL THE PROGRAMS FLOW OF
:2278 CONTROL.
:2279
:2280 JMP [] -- TRANSFER TO NEW ADDRESS UNCONDITIONALLY.
:2281
:2282 JMP.OS [] -- TRANSFER TO RESULTANT ADDRESS OF LOGICAL OR OF ADDRESS
:2283 IN THE INSTRUCTION AND THE OS REGISTER UNCONDITIONALLY.
:2284
:2285 JMP.IF[] TO [] -- TRANSFER TO NEW ADDRESS ONLY IF JUMP
:2286 CONDITION IS MET.
:2287
:2288 JSR [] -- TRANSFER TO NEW ADDRESS UNCONDITIONALLY AND PUT
:2289 RETURN ADDRESS ON THE SEQUENCER STACK.
:2290
:2291 JSR.OS [] -- TRANSFER TO RESULTANT ADDRESS OF LOGICAL OR OF
:2292 ADDRESS IN THE INSTRUCTION AND THE OS REGISTER.
:2293 THE RETURN ADDRESS IS ALSO PUSHED ON THE SEQUENCER STACK.
:2294
:2295 RETURN -- TRANSFER TO ADDRESS ON TOP OF THE SEQUENCER STACK AND
:2296 POP THE STACK.
:2297
:2298 RETURN+1 -- TRANSFER TO ADDRESS ON TOP OF THE SEQUENCER
:2299 STACK PLUS ONE AND POP THE STACK.
:2300
:2301 RETURN+1.IF(MEM.REF.OK) -- IF THE MOST RECENT MEMORY REQUEST HAD
:2302 NO ERROR THEN TRANSFER TO ADDRESS ON TOP OF THE SEQUENCER
:2303 STACK PLUS ONE AND POP THE STACK. IF THERE WAS AN
:2304 ERROR THEN SKIP OVER THE ENEXT INSTRUCTION.
:2305
:2306 LOOP.IF(NEQ) -- IF THE ALU Z-BIT IS ZERO (LAST VALUE CALCULATED
:2307 WHEN THE ALU.CC'S WERE SET DID NOT RESULT AS ZERO)
:2308 THEN TRANSFER TO THE ADDRESS ON TOP OF THE SEQUENCER
:2309 STACK. THE STACK IS NOT POPPED. IF THE ALU Z-BIT
:2310 IS ONE THEN CONTINUE WITH THE NEXT INSTRUCTION.
:2311 (UPC+1) AND THE STACK IS POPPED.
:2312
:2313 LOOP.IF(GEQU) -- IF THE ALU C-BIT IS ONE (LAST VALUE CALCULATED WHEN
:2314 THE ALU.CC'S WERE SET WAS GREATER THAN OR EQUAL TO ZERO)
:2315 THEN TRANSFER TO THE ADDRESS ON TOP OF THE SEQUENCER
:2316 STACK. THE STACK IS NOT POPPED. IF THE ALU C-BIT
:2317 IS ZERO THEN CONTINUE WITH THE NEXT INSTRUCTION
:2318 (UPC+1) AND THE STACK IS POPPED.
:2319
:2320 LOOP.IF(NO INTERRUPT AND NO SYNC) -- IF THE INTERRUPT CONDITION
:2321 IS NOT TRUE AND THE ACCELERATOR SYNC CONDITION IS NOT
:2322 TRUE THEN TRANSFER TO THE ADDRESS ON THE TOP OF THE SEQUENCER
:2323 STACK. THE STACK IS NOT POPPED. IF THE INTERRUPT
:2324 CONDITION IS TRUE THEN CONTINUE WITH THE NEXT INSTRUCTION
:2325 (UPC+1) AND THE SEQUENCER STACK IS NOT POPPED. IF THE ACCELERATOR
:2326 SYNC IS TRUE THEN CONTINUE WITH THE NEXT INSTRUCTION
:2327 (UPC+1) AND THE SEQUENCER STACK IS POPPED.
:2328

```

```

:2329 : SKIP.IF[] -- IF THE SKIP CONDITION IS SATISFIED THEN TRANSFER
:2330 : TO UPC+2 ELSE TRANSFER TO UPC+1.
:2331 :
:2332 :
:2333 JMP [] 'OPC2/JUMP,JUMP.ADRS/@1,JCTL/JUMP''
:2334 JMP.OS [] 'OPC2/JUMP,JUMP.ADRS/@1,OS/WITH.OS,JCTL/JUMP''
:2335 JMP.IF[] TO [] 'OPC2/JUMP,JCTL/@1,JUMP.ADRS/@2''
:2336 JSR [] 'OPC2/JUMP,JCTL/JSR,JUMP.ADRS/@1''
:2337 JSR.OS [] 'OPC2/JUMP,JCTL/JSR,JUMP.ADRS/@1,OS/WITH.OS''
:2338 :
:2339 RETURN 'SCTL/RETURN''
:2340 RETURN+1 'SCTL/RETURN+1''
:2341 RETURN+1.IF(MEM.REF.OK) 'SCTL/RET.NOERR''
:2342 :
:2343 LOOP.IF(NEQ) 'SCTL/JMP.Z.CLR''
:2344 LOOP.IF(GEQU) 'SCTL/JMP.C.SET''
:2345 LOOP.IF(NO INTERRUPT AND NO SYNC) ELSE SKIP.IF(SYNC) 'SCTL/LOOP.IF.NO.I.AND.SYNC''
:2346 :
:2347 SKIP.IF[] 'SCTL/@1''
:2348 SKIP 'SCTL/SKIP''

```

```

:2349 .PAGE
:2350 :
:2351 : INSTRUCTION STREAM MACROS
:2352 :
:2353 : * THIS MACRO IS NOT FOR GENERAL USE!!! *
:2354 : * THIS IS ONLY FOR USE IN THE FOLLOWING MACROS *
:2355 DEC[] 'OPC2/DECODE,DEC.ADRS/<.SHIFT[<JUMP.ADRS/@1>,-8]>'
:2356
:2357 DECODE.SPEC ADRS[] 'DEC[@1],IFUNC/SPEC,BPC/NOP,CM.HI/LOW.BYTE,IB.REQ/IB.REQ,LD.OS/LOAD.OS,R.DST/ENAB,OPC.SPEC/S
:2358 DECODE.ASRC ADRS[] 'DEC[@1],IFUNC/ASRC,BPC/NOP,CM.HI/LOW.BYTE,IB.REQ/IB.REQ,LD.OS/LOAD.OS,R.DST/ENAB,OPC.SPEC/A
:2359 DECODE.ESRC ADRS[] 'DEC[@1],IFUNC/ESRC,BPC/NOP,CM.HI/LOW.BYTE,IB.REQ/IB.REQ,LD.OS/LOAD.OS,R.DST/ENAB,OPC.SPEC/E
:2360 DECODE.VSRC ADRS[] 'DEC[@1],IFUNC/VSRC,BPC/NOP,CM.HI/LOW.BYTE,IB.REQ/IB.REQ,LD.OS/LOAD.OS,R.DST/ENAB,OPC.SPEC/V
:2361 DECODE.FLOAT ADRS[] 'DEC[@1],IFUNC/FLOAT,BPC/NOP,CM.HI/LOW.BYTE,IB.REQ/IB.REQ,LD.OS/LOAD.OS,R.DST/ENAB,OPC.SPEC/
:2362 DECODE.OPC1 ADRS[] 'DEC[@1],IFUNC/VAX.IRD,R.DST/NOP,CM.HI/LOW.BYTE,IB.REQ/IB.REQ,OPC.SPEC/VAX.IRD'
:2363 EXEC.DISPATCH ADRS[] 'DEC[@1],IFUNC/VAX.EXEC,IB.REQ/NOP,R.DST/NOP,CM.HI/LOW.BYTE,JCTL/JSR,OPC.SPEC/VAX.EXEC'
:2364
:2365 DECODE.CM.OPC ADRS[] 'DEC[@1],IFUNC/CM.IRD,CM.HI/HI.BYTE,OPC.SPEC/CM.IRD,LD.OS/LOAD.OS'
:2366 DECODE.CM.DST ADRS[] 'DEC[@1],IFUNC/CM.DST,OPC.SPEC/CM.DST,LD.OS/LOAD.OS,R.DST/ENAB'
:2367 DECODE.CM.SINGLE ADRS[] 'DEC[@1],IFUNC/CM.SINGLE,OPC.SPEC/CM.SINGLE'
:2368 CM.EXEC ADRS[] 'DEC[@1],IFUNC/CM.EXEC,JCTL/JUMP,OPC.SPEC/CM.EXEC,LD.OS/LOAD.OS'
:2369
:2370 SAVE.PC WR0 'BPC/SAVE.PC'
:2371 SAVE.PC WR4 'BPC/SAVE.PC'
:2372 SAVE.CM.PC WR0 'BPC/SAVE.PC'
:2373 SAVE.CM.PC WR4 'BPC/SAVE.PC'
:2374
:2375 :
:2376 : THESE SKIP CONDITIONS ARE USED FOR THE DECODE MICRO INSTRUCTION
:2377 :
:2378 SKIP.IF(IBM VALID) 'JCTL/NO.JUMP.TST'
:2379 JMP.IF(IBM VALID) 'JCTL/JUMP.VALID'
:2380 JSR.IF(IBM VALID) 'JCTL/JSR.VALID'
:2381 :
:2382 : THESE SKIP CONDITIONS ARE TO BE USED WITH THE COMPATIBILITY MODE DECODE
:2383 : MACROS.
:2384 :
:2385 JMP.TO [] 'JCTL/JUMP,DEC[@1]'
:2386 JSR.TO [] 'JCTL/JSR,DEC[@1]'
:2387 .BIN
    
```

```

:2388 .PAGE
:2389 .TOC 'FIELD DEFINITIONS FOR DATA PATH CONTROL MICRO WORD VER 00.01'
:2390
:2391 ; THIS SECTION IS FOR THE DATA PATH CONTROL ROMS
:2392 ;
:2393
:2394 .CCODE
:2395 SDP/=<8:0>,,ADDRESS,,VALIDITY=0
:2396
:2397 ; THIS FIELD CORRESPONDS TO THE 2901 ALU FUNCTION CONTROL PINS ... 15,14,13
:2398
:2399 ALU/=<2:0>,,DEFAULT=<ALU/R.OR.S>
:2400
:2401 R.PLUS.S=0 ; ADD R WITH S
:2402 S.MINUS.R=1 ; SUBTRACT R FROM S
:2403 R.MINUS.S=2 ; SUBTRACT S FROM R
:2404 R.OR.S=3 ; OR R WITH S
:2405 R.AND.S=4 ; AND R WITH S
:2406 NOTR.AND.S=5 ; COMPLEMENT R THEN AND WITH S
:2407 R.XOR.S=6 ; XOR R WITH S
:2408 R.XNOR.S=7 ; XOR R WITH S THEN COMPLEMENT THE RESULT
:2409 SRC/=<8:6>,,DEFAULT=<SRC/D.0>
:2410
:2411 O.Q=2 ; RMX=0 SMX=Q
:2412 O.B=3 ; RMX=0 SMX=B
:2413 A.Q=0 ; RMX=A SMX=Q
:2414 A.B=01 ; RMX=A SMX=B
:2415 D.Q=06 ; RMX=D SMX=Q
:2416 D.O=7 ; RMX=D SMX=O
:2417 O.A=4 ; RMX=O SMX=A
:2418 D.A=5 ; RMX=D SMX=A
:2419
:2420 ; THIS FIELD CORRESPONDES TO THE 2901 ALU DESTINATION
:2421 ; CONTROL PINS 18, 17 AND 16.
:2422
:2423 DST/=<5:3>,,DEFAULT=<DST/NOP>
:2424
:2425 LOAD.Q=0 ; LOAD Q-REGISTER
:2426 NOP=1 ; HOLD ALL REGISTERS
:2427 WRITE.B.A=2 ; WRITE RAM B-PORT AND OUTPUT RAM A-PORT
:2428 WRITE.B.F=3 ; WRITE RAM B-PORT AND OUTPUT ALU
:2429 RSHF.RAM.Q=4 ; RIGHT SHIFT RAM AND Q
:2430 RSHF.RAM=5 ; RIGHT SHIFT RAM
:2431 LSHF.RAM.Q=6 ; LEFT SHIFT RAM AND Q
:2432 LSHF.RAM=7 ; LEFT SHIFT RAM
:2433
:2434 CIN/=<9>,,DEFAULT=0
:2435
:2436 NO.CIN=0 ; NO CARRY IN TO ALU
:2437 CIN=1 ; CARRY IN TO ALU
:2438
:2439
:2440

```

```

:2441 .PAGE
:2442 .TOC "CONTROL ROM CONTENTS VER 00.01"
:2443 .SEQUENTIAL
:2444 : THESE ARE THE DATA PATH CONTROL ROM CONTENTS
:2445 :
:2446 : THE FOLLOWING LOCATIONS ARE USED BY THE DECODE.IS MICRO INSTRUCTION
:2447 :
:2448 : THIS INSTRUCTION WILL DO PC+1 OPERATIONS
:2449 :
:2450 : BACK UP PC IN 2901 RAM
:2451 000:
:2452 DECODE.IS:
C 0000. 3D8 :2453 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 0001. 3D8 :2454 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 0002. 3D8 :2455 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 0003. 3D8 :2456 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 0004. 3D8 :2457 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 0005. 3D8 :2458 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 0006. 3D8 :2459 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 0007. 3D8 :2460 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 0008. 3D8 :2461 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 0009. 3D8 :2462 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 000A. 3D8 :2463 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 000B. 3D8 :2464 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 000C. 3D8 :2465 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 000D. 3D8 :2466 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 000E. 3D8 :2467 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 000F. 3D8 :2468 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
:2469 :
:2470 : DON'T BACK UP PC
:2471 :
C 0010. 3C8 :2472 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 0011. 3C8 :2473 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 0012. 3C8 :2474 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 0013. 3C8 :2475 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 0014. 3C8 :2476 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 0015. 3C8 :2477 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 0016. 3C8 :2478 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 0017. 3C8 :2479 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 0018. 3C8 :2480 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 0019. 3C8 :2481 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 001A. 3C8 :2482 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 001B. 3C8 :2483 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 001C. 3C8 :2484 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 001D. 3C8 :2485 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 001E. 3C8 :2486 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 001F. 3C8 :2487 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN

```

:2488 .PAGE
 :2489 : THESE ADDRESSES ARE USED BY THE 'JUMP' MICRO INSTRUCTION
 :2490 :
 :2491 : THE FOLLOWING CONTROL CODES WILL INSURE THAT NO DATA IS DESTROYED WITHIN
 :2492 : THE 2901 DURING JUMP OR JSR OPERATIONS.
 :2493 :

:2494 020:
 :2495 JUMP:

C 0020.	3CB	:2496	SRC/D.O,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0021.	3CB	:2497	SRC/D.O,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0022.	3CB	:2498	SRC/D.O,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0023.	3CB	:2499	SRC/D.O,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0024.	3CB	:2500	SRC/D.O,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0025.	3CB	:2501	SRC/D.O,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0026.	3CB	:2502	SRC/D.O,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0027.	3CB	:2503	SRC/D.O,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0028.	3CB	:2504	SRC/D.O,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0029.	3CB	:2505	SRC/D.O,ALU/R.OR.S,DST/NOP,CIN/CIN
C 002A.	3CB	:2506	SRC/D.O,ALU/R.OR.S,DST/NOP,CIN/CIN
C 002B.	3CB	:2507	SRC/D.O,ALU/R.OR.S,DST/NOP,CIN/CIN
C 002C.	3CB	:2508	SRC/D.O,ALU/R.OR.S,DST/NOP,CIN/CIN
C 002D.	3CB	:2509	SRC/D.O,ALU/R.OR.S,DST/NOP,CIN/CIN
C 002E.	3CB	:2510	SRC/D.O,ALU/R.OR.S,DST/NOP,CIN/CIN
C 002F.	3CB	:2511	SRC/D.O,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0030.	3CB	:2512	SRC/D.O,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0031.	3CB	:2513	SRC/D.O,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0032.	3CB	:2514	SRC/D.O,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0033.	3CB	:2515	SRC/D.O,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0034.	3CB	:2516	SRC/D.O,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0035.	3CB	:2517	SRC/D.O,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0036.	3CB	:2518	SRC/D.O,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0037.	3CB	:2519	SRC/D.O,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0038.	3CB	:2520	SRC/D.O,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0039.	3CB	:2521	SRC/D.O,ALU/R.OR.S,DST/NOP,CIN/CIN
C 003A.	3CB	:2522	SRC/D.O,ALU/R.OR.S,DST/NOP,CIN/CIN
C 003B.	3CB	:2523	SRC/D.O,ALU/R.OR.S,DST/NOP,CIN/CIN
C 003C.	3CB	:2524	SRC/D.O,ALU/R.OR.S,DST/NOP,CIN/CIN
C 003D.	3CB	:2525	SRC/D.O,ALU/R.OR.S,DST/NOP,CIN/CIN
C 003E.	3CB	:2526	SRC/D.O,ALU/R.OR.S,DST/NOP,CIN/CIN
C 003F.	3CB	:2527	SRC/D.O,ALU/R.OR.S,DST/NOP,CIN/CIN

:2528 .PAGE
 :2529 : THESE LOCATIONS ARE USED BY THE MISC' MICRO INSTRUCTION
 :2530 :
 :2531 : THE FOLLOWING CONTROL CODES WILL INSURE THAT NO DATA IS DESTROYED WITHIN
 :2532 : THE 2901 DURING MISC INSTRUCTION EXECUTION.
 :2533 :

:2534 040:
 :2535 MISC:

C 0040, 313	:2536	SRC/O.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0041, 313	:2537	SRC/O.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0042, 313	:2538	SRC/O.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0043, 313	:2539	SRC/O.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0044, 313	:2540	SRC/O.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0045, 313	:2541	SRC/O.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0046, 313	:2542	SRC/O.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0047, 313	:2543	SRC/O.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0048, 313	:2544	SRC/O.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0049, 313	:2545	SRC/O.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 004A, 313	:2546	SRC/O.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 004B, 313	:2547	SRC/O.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 004C, 313	:2548	SRC/O.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 004D, 313	:2549	SRC/O.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 004E, 313	:2550	SRC/O.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 004F, 313	:2551	SRC/O.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0050, 313	:2552	SRC/O.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0051, 313	:2553	SRC/O.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0052, 313	:2554	SRC/O.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0053, 313	:2555	SRC/O.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0054, 313	:2556	SRC/O.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0055, 313	:2557	SRC/O.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0056, 313	:2558	SRC/O.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0057, 313	:2559	SRC/O.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0058, 313	:2560	SRC/O.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0059, 313	:2561	SRC/O.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 005A, 313	:2562	SRC/O.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 005B, 313	:2563	SRC/O.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 005C, 313	:2564	SRC/O.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 005D, 313	:2565	SRC/O.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 005E, 313	:2566	SRC/O.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 005F, 313	:2567	SRC/O.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN

C	O
C	O
C	O
C	O
C	O
C	O
C	O
C	O

```

:2607 .PAGE
:2608 : THESE ARE THE ADDRESSES FOR THE EXTENDED MICRO INSTRUCTION
:2609 : ADDRESSES START AT 80-FF
:2610 080:
:2611
:2612 WR.PLUS.O.TO.WR:
C 0080, 118 :2613 SRC/O.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
:2614
:2615 WR.INC.WR:
C 0081, 318 :2616 SRC/O.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
:2617
:2618 WR.NEG.WR:
C 0082, 31A :2619 SRC/O.A,ALU/R.MINUS.S,DST/WRITE.B.F,CIN/CIN
:2620
:2621 WR.COM.WR:
C 0083, 31F :2622 SRC/O.A,ALU/R.XNOR.S,DST/WRITE.B.F,CIN/CIN
:2623
:2624 WR.DEC.WR:
C 0084, 119 :2625 SRC/O.A,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/NO.CIN
:2626
:2627 FILL85:
C 0085, 10B :2628 SRC/O.A
:2629
:2630 MOV.Q.WR:
C 0086, 29B :2631 SRC/O.Q,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
:2632
:2633 FILL87:
C 0087, 08B :2634 SRC/O.Q
:2635
:2636 COND.ADD&SHIFT:
C 0088, 060 :2637 SRC/A.B,ALU/R.PLUS.S,DST/RSHF.RAM.Q,CIN/NO.CIN
:2638
:2639 COND.SUB&SHIFT:
C 0089, 261 :2640 SRC/A.B,ALU/S.MINUS.R,DST/RSHF.RAM.Q,CIN/CIN
:2641
:2642 FILL8A:
C 008A, 04B :2643 SRC/A.B
:2644
:2645 SHL2:
C 008B, 078 :2646 SRC/A.B,ALU/R.PLUS.S,DST/LSHF.RAM,CIN/NO.CIN
:2647
:2648 ASHRC:
C 008C, 0E3 :2649 SRC/O.B,ALU/R.OR.S,DST/RSHF.RAM.Q,CIN/NO.CIN
:2650
:2651 ASHR:
C 008D, 2EB :2652 SRC/O.B,ALU/R.OR.S,DST/RSHF.RAM,CIN/CIN
:2653
:2654 ASHLC:
C 008E, 2F3 :2655 SRC/O.B,ALU/R.OR.S,DST/LSHF.RAM.Q,CIN/CIN
:2656
:2657 ASHL:
C 008F, 2FB :2658 SRC/O.B,ALU/R.OR.S,DST/LSHF.RAM,CIN/CIN
:2659
:2660 Q.PLUS.O:
C 0090, 088 :2661 SRC/O.Q,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
:2662
:2663 FILL91:
C 0091, 08B :2664 SRC/O.Q
:2665
:2666 WR.CMP.Q:
C 0092, 20A :2667 SRC/A.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
:2668
:2669 Q.CMP.WR:
C 0093, 209 :2670 SRC/A.Q,ALU/S.MINUS.R,DST/NOP,CIN/CIN
:2671
:2672 DEC.Q:
C 0094, 081 :2673 SRC/O.Q,ALU/S.MINUS.R,DST/LOAD.Q,CIN/NO.CIN
:2674
:2675 INC.Q:
C 0095, 280 :2676 SRC/O.Q,ALU/R.PLUS.S,DST/LOAD.Q,CIN/CIN
:2677
:2678 NEG.Q:
    
```

22-ENKCF-1.4	:	ENKCF.MIC	CONTROL ROM CONTENTS	C 6	15-MAR-83	Fiche 1 Frame C6	Sequence 67	Page 61
:	:	ENKCF.MCR	MICRO2 1M(01)	15-MAR-83	11:46:22	ENKCF Micro-diagnostic for IDC module	Rev 01.40	
:	:	ENKCF.MIC	CONTROL ROM CONTENTS			VER 00.01		

C 0096, 282	:2662	SRC/O.Q,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
	:2663	COM.Q:
C 0097, 287	:2664	SRC/O.Q,ALU/R.XNOR.S,DST/LOAD.Q,CIN/CIN
	:2665	
	:2666	WR.BIT.WR:
C 0098, 04C	:2667	SRC/A.B,ALU/R.AND.S,DST/NOP,CIN/NO.CIN
	:2668	WR.CMP.WR:
C 0099, 24A	:2669	SRC/A.B,ALU/R.MINUS.S,DST/NOP,CIN/CIN
	:2670	FILL9A:
C 009A, 04B	:2671	SRC/A.B
	:2672	WR.PLUS.WR+1:
C 009B, 258	:2673	SRC/A.B,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
	:2674	
	:2675	FILL9C:
C 009C, 04B	:2676	SRC/A.B
	:2677	FILL9D:
C 009D, 04B	:2678	SRC/A.B
	:2679	FILL9E:
C 009E, 0CB	:2680	SRC/O.B
	:2681	FILL9F:
C 009F, 0CB	:2682	SRC/O.B
	:2683	
	:2684	WR.PLUS.Q:
C 00A0, 000	:2685	SRC/A.Q,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
	:2686	WR.FROM.Q:
C 00A1, 201	:2687	SRC/A.Q,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
	:2688	Q.FROM.WR.Q:
C 00A2, 202	:2689	SRC/A.Q,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
	:2690	WR.OR.Q:
C 00A3, 203	:2691	SRC/A.Q,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
	:2692	
	:2693	WR.AND.Q:
C 00A4, 004	:2694	SRC/A.Q,ALU/R.AND.S,DST/LOAD.Q,CIN/NO.CIN
	:2695	WR.MASK.Q:
C 00A5, 205	:2696	SRC/A.Q,ALU/NOTR.AND.S,DST/LOAD.Q,CIN/CIN
	:2697	WR.XOR.Q:
C 00A6, 206	:2698	SRC/A.Q,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
	:2699	FILLA7:
C 00A7, 00B	:2700	SRC/A.Q
	:2701	
	:2702	WR.PLUS.WR.Q:
C 00A8, 040	:2703	SRC/A.B,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
	:2704	WR.FROM.WR.Q:
C 00A9, 241	:2705	SRC/A.B,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
	:2706	FILLAA:
C 00AA, 04B	:2707	SRC/A.B
	:2708	WR.OR.WR.Q:
C 00AB, 243	:2709	SRC/A.B,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
	:2710	
	:2711	WR.AND.WR.Q:
C 00AC, 044	:2712	SRC/A.B,ALU/R.AND.S,DST/LOAD.Q,CIN/NO.CIN
	:2713	WR.MASK.WR.Q:
C 00AD, 245	:2714	SRC/A.B,ALU/NOTR.AND.S,DST/LOAD.Q,CIN/CIN
	:2715	WR.XOR.WR.Q:
C 00AE, 246	:2716	SRC/A.B,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN

```

C 00AF, 04B :2717 FILLAF:
               :2718 SRC/A.B
               :2719
C 00B0, 018 :2720 Q.PLUS.WR:
               :2721 SRC/A.Q,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
               :2722 WR.FROM.Q.WR:
C 00B1, 219 :2723 SRC/A.Q,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
               :2724 Q.FROM.WR:
C 00B2, 21A :2725 SRC/A.Q,ALU/R.MINUS.S,DST/WRITE.B.F,CIN/CIN
               :2726 Q.OR.WR:
C 00B3, 21B :2727 SRC/A.Q,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
               :2728
               :2729 Q.AND.WR:
C 00B4, 01C :2730 SRC/A.Q,ALU/R.AND.S,DST/WRITE.B.F,CIN/NO.CIN
               :2731 FILLB
C 00B5, 00B :2732 SRC/A.Q
               :2733 Q.XOR.WR:
C 00B6, 21E :2734 SRC/A.Q,ALU/R.XOR.S,DST/WRITE.B.F,CIN/CIN
               :2735 Q.BIT.WR:
C 00B7, 20C :2736 SRC/A.Q,ALU/R.AND.S,DST/NOP,CIN/CIN
               :2737
               :2738 WR.PLUS.WR:
C 00B8, 05B :2739 SRC/A.B,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
               :2740 WR.FROM.WR:
C 00B9, 259 :2741 SRC/A.B,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
               :2742 WR.FROM.WR.WR:
C 00BA, 25A :2743 SRC/A.B,ALU/R.MINUS.S,DST/WRITE.B.F,CIN/CIN
               :2744 WR.OR.WR:
C 00BB, 25B :2745 SRC/A.B,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
               :2746
               :2747 WR.FROM.WR-1:
C 00BC, 059 :2748 SRC/A.B,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/NO.CIN
               :2749 WR.MASK.WR:
C 00BD, 25D :2750 SRC/A.B,ALU/NOTR.AND.S,DST/WRITE.B.F,CIN/CIN
               :2751 WR.XOR.WR:
C 00BE, 25E :2752 SRC/A.B,ALU/R.XOR.S,DST/WRITE.B.F,CIN/CIN
               :2753 WR.AND.WR:
C 00BF, 25C :2754 SRC/A.B,ALU/R.AND.S,DST/WRITE.B.F,CIN/CIN
               :2755
  
```

```

:2756 .PAGE
:2757 : THIS IS THE DP CODES FOR THE 'MOVE' MICRO INSTRUCTION
:2758 :
:2759 0C0:
:2760
:2761 MOV.MEM.WR:
C 00C0, 1D3 :2762 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
C 00C1, 1D3 :2763 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
C 00C2, 1D3 :2764 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
C 00C3, 1D3 :2765 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
C 00C4, 1D3 :2766 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
C 00C5, 1D3 :2767 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
C 00C6, 1D3 :2768 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
C 00C7, 1D3 :2769 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
:2770
:2771 MOV.LS.MEM:
C 00C8, 1CB :2772 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00C9, 1CB :2773 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00CA, 1CB :2774 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00CB, 1CB :2775 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00CC, 1CB :2776 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00CD, 1CB :2777 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00CE, 1CB :2778 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00CF, 1CB :2779 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
:2780
:2781 MOV.XWR.Q:
C 00D0, 0C3 :2782 SRC/O.B,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
C 00D1, 0C3 :2783 SRC/O.B,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
C 00D2, 0C3 :2784 SRC/O.B,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
C 00D3, 0C3 :2785 SRC/O.B,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
C 00D4, 0C3 :2786 SRC/O.B,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
C 00D5, 0C3 :2787 SRC/O.B,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
C 00D6, 0C3 :2788 SRC/O.B,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
C 00D7, 0C3 :2789 SRC/O.B,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
:2790
:2791 MOV.LS.WR:
C 00D8, 1DB :2792 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/NO.CIN
C 00D9, 1DB :2793 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/NO.CIN
C 00DA, 1DB :2794 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/NO.CIN
C 00DB, 1DB :2795 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/NO.CIN
C 00DC, 1DB :2796 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/NO.CIN
C 00DD, 1DB :2797 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/NO.CIN
C 00DE, 1DB :2798 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/NO.CIN
C 00DF, 1DB :2799 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/NO.CIN
:2800
    
```

```

:2801 .PAGE
:2802 MOV.MEM.LS:
C 00E0, 1CB :2803 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00E1, 1CB :2804 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00E2, 1CB :2805 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00E3, 1CB :2806 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00E4, 1CB :2807 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00E5, 1CB :2808 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00E6, 1CB :2809 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00E7, 1CB :2810 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
:2811 MOV.ACC.LS:
C 00E8, 1CB :2812 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00E9, 1CB :2813 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00EA, 1CB :2814 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00EB, 1CB :2815 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00EC, 1CB :2816 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00ED, 1CB :2817 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00EE, 1CB :2818 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00EF, 1CB :2819 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
:2820
:2821
:2822 WR.PLUS.OS:
C 00F0, 148 :2823 SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 00F1, 148 :2824 SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 00F2, 148 :2825 SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 00F3, 148 :2826 SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 00F4, 148 :2827 SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 00F5, 148 :2828 SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 00F6, 148 :2829 SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 00F7, 148 :2830 SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
:2831
:2832 MOV.WR.LS:
C 00F8, 10B :2833 SRC/O.A,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00F9, 10B :2834 SRC/O.A,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00FA, 10B :2835 SRC/O.A,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00FB, 10B :2836 SRC/O.A,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00FC, 10B :2837 SRC/O.A,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00FD, 10B :2838 SRC/O.A,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00FE, 10B :2839 SRC/O.A,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00FF, 10B :2840 SRC/O.A,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
:2841
    
```

```

:2842 .PAGE
:2843 : THIS GROUP OF INSTRUCTIONS IS FOR THE 'BASIC' MICRO INSTRUCTION
:2844 : THIS USES ADDRESSES 100-1FF
:2845
:2846 100:
:2847 LS.PLUS.WR:
C 0100, 158 SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
C 0101, 158 SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
C 0102, 158 SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
C 0103, 158 SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
:2851
:2852 LS.FROM.WR:
C 0104, 359 SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
C 0105, 359 SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
C 0106, 359 SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
C 0107, 359 SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
:2856
:2857 LS.AND.WR:
C 0108, 35C SRC/D.A,ALU/R.AND.S,DST/WRITE.B.F,CIN/CIN
C 0109, 35C SRC/D.A,ALU/R.AND.S,DST/WRITE.B.F,CIN/CIN
C 010A, 35C SRC/D.A,ALU/R.AND.S,DST/WRITE.B.F,CIN/CIN
C 010B, 35C SRC/D.A,ALU/R.AND.S,DST/WRITE.B.F,CIN/CIN
:2861
:2862 LS.XOR.WR:
C 010C, 35E SRC/D.A,ALU/R.XOR.S,DST/WRITE.B.F,CIN/CIN
C 010D, 35E SRC/D.A,ALU/R.XOR.S,DST/WRITE.B.F,CIN/CIN
C 010E, 35E SRC/D.A,ALU/R.XOR.S,DST/WRITE.B.F,CIN/CIN
C 010F, 35E SRC/D.A,ALU/R.XOR.S,DST/WRITE.B.F,CIN/CIN
:2866
:2867
:2868 LS.FROM.WR-1:
C 0110, 159 SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/NO.CIN
C 0111, 159 SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/NO.CIN
C 0112, 159 SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/NO.CIN
C 0113, 159 SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/NO.CIN
:2872
:2873 LS.MASK.WR:
C 0114, 35D SRC/D.A,ALU/NOTR.AND.S,DST/WRITE.B.F,CIN/CIN
C 0115, 35D SRC/D.A,ALU/NOTR.AND.S,DST/WRITE.B.F,CIN/CIN
C 0116, 35D SRC/D.A,ALU/NOTR.AND.S,DST/WRITE.B.F,CIN/CIN
C 0117, 35D SRC/D.A,ALU/NOTR.AND.S,DST/WRITE.B.F,CIN/CIN
:2877
:2878 LS.PLUS.WR+1:
C 0118, 358 SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 0119, 358 SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 011A, 358 SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 011B, 358 SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
:2882
:2883 LS.OR.WR:
C 011C, 35B SRC/D.A,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 011D, 35B SRC/D.A,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 011E, 35B SRC/D.A,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 011F, 35B SRC/D.A,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
:2887
:2888
:2889 WR.PLUS.LS.Q:
C 0120, 140 SRC/D.A,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
C 0121, 140 SRC/D.A,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
C 0122, 140 SRC/D.A,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
C 0123, 140 SRC/D.A,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
:2893
:2894 LS.FROM.WR.Q:
C 0124, 341 SRC/D.A,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
C 0125, 341 SRC/D.A,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
:2896
    
```


C 0126, 341	:2897	SRC/D.A,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
C 0127, 341	:2898	SRC/D.A,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
	:2899	WR.FROM.LS.Q:
C 0128, 342	:2900	SRC/D.A,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
C 0129, 342	:2901	SRC/D.A,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
C 012A, 342	:2902	SRC/D.A,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
C 012B, 342	:2903	SRC/D.A,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
	:2904	WR.OR.LS.Q:
C 012C, 343	:2905	SRC/D.A,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
C 012D, 343	:2906	SRC/D.A,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
C 012E, 343	:2907	SRC/D.A,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
C 012F, 343	:2908	SRC/D.A,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
	:2909	
	:2910	WR.AND.LS.Q:
C 0130, 144	:2911	SRC/D.A,ALU/R.AND.S,DST/LOAD.Q,CIN/NO.CIN
C 0131, 144	:2912	SRC/D.A,ALU/R.AND.S,DST/LOAD.Q,CIN/NO.CIN
C 0132, 144	:2913	SRC/D.A,ALU/R.AND.S,DST/LOAD.Q,CIN/NO.CIN
C 0133, 144	:2914	SRC/D.A,ALU/R.AND.S,DST/LOAD.Q,CIN/NO.CIN
	:2915	WR.XOR.LS.Q:
C 0134, 346	:2916	SRC/D.A,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
C 0135, 346	:2917	SRC/D.A,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
C 0136, 346	:2918	SRC/D.A,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
C 0137, 346	:2919	SRC/D.A,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
	:2920	LS.CMP.WR:
C 0138, 34A	:2921	SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 0139, 34A	:2922	SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 013A, 34A	:2923	SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 013B, 34A	:2924	SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/CIN
	:2925	WR.CMP.LS:
C 013C, 349	:2926	SRC/D.A,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 013D, 349	:2927	SRC/D.A,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 013E, 349	:2928	SRC/D.A,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 013F, 349	:2929	SRC/D.A,ALU/S.MINUS.R,DST/NOP,CIN/CIN
	:2930	
	:2931	LS.PLUS.Q:
C 0140, 180	:2932	SRC/D.Q,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
C 0141, 180	:2933	SRC/D.Q,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
C 0142, 180	:2934	SRC/D.Q,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
C 0143, 180	:2935	SRC/D.Q,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
	:2936	LS.FROM.Q:
C 0144, 381	:2937	SRC/D.Q,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
C 0145, 381	:2938	SRC/D.Q,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
C 0146, 381	:2939	SRC/D.Q,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
C 0147, 381	:2940	SRC/D.Q,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
	:2941	Q.FROM.LS.Q:
C 0148, 382	:2942	SRC/D.Q,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
C 0149, 382	:2943	SRC/D.Q,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
C 014A, 382	:2944	SRC/D.Q,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
C 014B, 382	:2945	SRC/D.Q,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
	:2946	LS.OR.Q:
C 014C, 383	:2947	SRC/D.Q,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
C 014D, 383	:2948	SRC/D.Q,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
C 014E, 383	:2949	SRC/D.Q,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
C 014F, 383	:2950	SRC/D.Q,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
	:2951	

```

:2952 LS.MASK.Q:
C 0150, 185 :2953 SRC/D.Q,ALU/NOTR.AND.S,DST/LOAD.Q,CIN/NO.CIN
C 0151, 185 :2954 SRC/D.Q,ALU/NOTR.AND.S,DST/LOAD.Q,CIN/NO.CIN
C 0152, 185 :2955 SRC/D.Q,ALU/NOTR.AND.S,DST/LOAD.Q,CIN/NO.CIN
C 0153, 185 :2956 SRC/D.Q,ALU/NOTR.AND.S,DST/LOAD.Q,CIN/NO.CIN
:2957 LS.XOR.Q:
C 0154, 386 :2958 SRC/D.Q,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
C 0155, 386 :2959 SRC/D.Q,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
C 0156, 386 :2960 SRC/D.Q,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
C 0157, 386 :2961 SRC/D.Q,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
:2962 LS.AND.Q:
C 0158, 384 :2963 SRC/D.Q,ALU/R.AND.S,DST/LOAD.Q,CIN/CIN
C 0159, 384 :2964 SRC/D.Q,ALU/R.AND.S,DST/LOAD.Q,CIN/CIN
C 015A, 384 :2965 SRC/D.Q,ALU/R.AND.S,DST/LOAD.Q,CIN/CIN
C 015B, 384 :2966 SRC/D.Q,ALU/R.AND.S,DST/LOAD.Q,CIN/CIN
:2967 LS.BIT.Q:
C 015C, 38C :2968 SRC/D.Q,ALU/R.AND.S,DST/NOP,CIN/CIN
C 015D, 38C :2969 SRC/D.Q,ALU/R.AND.S,DST/NOP,CIN/CIN
C 015E, 38C :2970 SRC/D.Q,ALU/R.AND.S,DST/NOP,CIN/CIN
C 015F, 38C :2971 SRC/D.Q,ALU/R.AND.S,DST/NOP,CIN/CIN
:2972
:2973 MOV.LS.Q:
C 0160, 1C3 :2974 SRC/D.Q,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
C 0161, 1C3 :2975 SRC/D.Q,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
C 0162, 1C3 :2976 SRC/D.Q,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
C 0163, 1C3 :2977 SRC/D.Q,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
:2978 WR.BIT.LS:
C 0164, 34C :2979 SRC/D.A,ALU/R.AND.S,DST/NOP,CIN/CIN
C 0165, 34C :2980 SRC/D.A,ALU/R.AND.S,DST/NOP,CIN/CIN
C 0166, 34C :2981 SRC/D.A,ALU/R.AND.S,DST/NOP,CIN/CIN
C 0167, 34C :2982 SRC/D.A,ALU/R.AND.S,DST/NOP,CIN/CIN
:2983 LS.CMP.Q:
C 0168, 38A :2984 SRC/D.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 0169, 38A :2985 SRC/D.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 016A, 38A :2986 SRC/D.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 016B, 38A :2987 SRC/D.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
:2988 Q.CMP.LS:
C 016C, 389 :2989 SRC/D.Q,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 016D, 389 :2990 SRC/D.Q,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 016E, 389 :2991 SRC/D.Q,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 016F, 389 :2992 SRC/D.Q,ALU/S.MINUS.R,DST/NOP,CIN/CIN
:2993
:2994 Q.PLUS.LS.TO.WR:
C 0170, 198 :2995 SRC/D.Q,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
C 0171, 198 :2996 SRC/D.Q,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
C 0172, 198 :2997 SRC/D.Q,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
C 0173, 198 :2998 SRC/D.Q,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
:2999 WRA.FROM.LS.WRB:
C 0174, 35A :3000 SRC/D.A,ALU/R.MINUS.S,DST/WRITE.B.F,CIN/CIN
C 0175, 35A :3001 SRC/D.A,ALU/R.MINUS.S,DST/WRITE.B.F,CIN/CIN
C 0176, 35A :3002 SRC/D.A,ALU/R.MINUS.S,DST/WRITE.B.F,CIN/CIN
C 0177, 35A :3003 SRC/D.A,ALU/R.MINUS.S,DST/WRITE.B.F,CIN/CIN
:3004 LS.NEG.WR:
C 0178, 3D9 :3005 SRC/D.Q,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
C 0179, 3D9 :3006 SRC/D.Q,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
    
```

C 017A, 3D9	:3007	SRC/D.0,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
C 017B, 3D9	:3008	SRC/D.0,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
	:3009	LS.COM.WR:
C 017C, 3DF	:3010	SRC/D.0,ALU/R.XNOR.S,DST/WRITE.B.F,CIN/CIN
C 017D, 3DF	:3011	SRC/D.0,ALU/R.XNOR.S,DST/WRITE.B.F,CIN/CIN
C 017E, 3DF	:3012	SRC/D.0,ALU/R.XNOR.S,DST/WRITE.B.F,CIN/CIN
C 017F, 3DF	:3013	SRC/D.0,ALU/R.XNOR.S,DST/WRITE.B.F,CIN/CIN

```

:3014 .PAGE
:3015
:3016 : THE FOLLOWING LOCATIONS HAVE THE LOCAL STORE AS THEIR DESTINATION.
:3017
:3018 : THIS FACT IS USED BY HARDWARE TO GENERATE LOCAL STORE WRITE PULSES
:3019
:3020 180:
:3021
:3022 LS.PLUS.WR.XCHG:
C 0180, 150 :3023 SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.A,CIN/NO.CIN
C 0181, 150 :3024 SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.A,CIN/NO.CIN
C 0182, 150 :3025 SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.A,CIN/NO.CIN
C 0183, 150 :3026 SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.A,CIN/NO.CIN
:3027 LS.FROM.WR.XCHG:
C 0184, 351 :3028 SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.A,CIN/CIN
C 0185, 351 :3029 SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.A,CIN/CIN
C 0186, 351 :3030 SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.A,CIN/CIN
C 0187, 351 :3031 SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.A,CIN/CIN
:3032 LS.AND.WR.XCHG:
C 0188, 354 :3033 SRC/D.A,ALU/R.AND.S,DST/WRITE.B.A,CIN/CIN
C 0189, 354 :3034 SRC/D.A,ALU/R.AND.S,DST/WRITE.B.A,CIN/CIN
C 018A, 354 :3035 SRC/D.A,ALU/R.AND.S,DST/WRITE.B.A,CIN/CIN
C 018B, 354 :3036 SRC/D.A,ALU/R.AND.S,DST/WRITE.B.A,CIN/CIN
:3037 LS.XOR.WR.XCHG:
C 018C, 356 :3038 SRC/D.A,ALU/R.XOR.S,DST/WRITE.B.A,CIN/CIN
C 018D, 356 :3039 SRC/D.A,ALU/R.XOR.S,DST/WRITE.B.A,CIN/CIN
C 018E, 356 :3040 SRC/D.A,ALU/R.XOR.S,DST/WRITE.B.A,CIN/CIN
C 018F, 356 :3041 SRC/D.A,ALU/R.XOR.S,DST/WRITE.B.A,CIN/CIN
:3042
:3043 SWAP.LS.WR:
C 0190, 1D3 :3044 SRC/D.O,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
C 0191, 1D3 :3045 SRC/D.O,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
C 0192, 1D3 :3046 SRC/D.O,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
C 0193, 1D3 :3047 SRC/D.O,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
:3048 CLR.LS:
C 0194, 3CC :3049 SRC/D.O,ALU/R.AND.S,DST/NOP,CIN/CIN
C 0195, 3CC :3050 SRC/D.O,ALU/R.AND.S,DST/NOP,CIN/CIN
C 0196, 3CC :3051 SRC/D.O,ALU/R.AND.S,DST/NOP,CIN/CIN
C 0197, 3CC :3052 SRC/D.O,ALU/R.AND.S,DST/NOP,CIN/CIN
:3053 MOV.Q.WR&WR.LS:
C 0198, 293 :3054 SRC/O.Q,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0199, 293 :3055 SRC/O.Q,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 019A, 293 :3056 SRC/O.Q,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 019B, 293 :3057 SRC/O.Q,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
:3058 WR.PLUS.Q.XCHG:
C 019C, 010 :3059 SRC/A.Q,ALU/R.PLUS.S,DST/WRITE.B.A,CIN/NO.CIN
C 019D, 010 :3060 SRC/A.Q,ALU/R.PLUS.S,DST/WRITE.B.A,CIN/NO.CIN
C 019E, 010 :3061 SRC/A.Q,ALU/R.PLUS.S,DST/WRITE.B.A,CIN/NO.CIN
C 019F, 010 :3062 SRC/A.Q,ALU/R.PLUS.S,DST/WRITE.B.A,CIN/NO.CIN
:3063
:3064 WR.FROM.LS-1:
C 01A0, 14A :3065 SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/NO.CIN
C 01A1, 14A :3066 SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/NO.CIN
C 01A2, 14A :3067 SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/NO.CIN
C 01A3, 14A :3068 SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/NO.CIN

```

ZZ-ENKCF-1.4 :
: ENKCF.MCR
: ENKCF.MIC

ENKCF.MIC

MICRO2 1M(01)
CONTROL ROM CONTENTS

CONTROL ROM CONTENTS

L 6
15-MAR-83

Fiche 1 Frame L6

Sequence 76

ENKCF Micro-diagnostic for IDC module
VER 00.01

Rev 01.40

Page 70

ZZ-
:
:

```
:3069 LS.FROM.WR.LS:
C 01A4, 349 :3070 SRC/D.A,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 01A5, 349 :3071 SRC/D.A,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 01A6, 349 :3072 SRC/D.A,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 01A7, 349 :3073 SRC/D.A,ALU/S.MINUS.R,DST/NOP,CIN/CIN
:3074 WR.PLUS.LS+1:
C 01A8, 348 :3075 SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 01A9, 348 :3076 SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 01AA, 348 :3077 SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 01AB, 348 :3078 SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/CIN
:3079 WR.FROM.LS:
C 01AC, 34A :3080 SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01AD, 34A :3081 SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01AE, 34A :3082 SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01AF, 34A :3083 SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/CIN
:3084
:3085 WR.PLUS.LS:
C 01B0, 148 :3086 SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 01B1, 148 :3087 SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 01B2, 148 :3088 SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 01B3, 148 :3089 SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
:3090 WR.OR.LS:
C 01B4, 34B :3091 SRC/D.A,ALU/R.OR.S,DST/NOP,CIN/CIN
C 01B5, 34B :3092 SRC/D.A,ALU/R.OR.S,DST/NOP,CIN/CIN
C 01B6, 34B :3093 SRC/D.A,ALU/R.OR.S,DST/NOP,CIN/CIN
C 01B7, 34B :3094 SRC/D.A,ALU/R.OR.S,DST/NOP,CIN/CIN
:3095 WR.AND.LS:
C 01B8, 34C :3096 SRC/D.A,ALU/R.AND.S,DST/NOP,CIN/CIN
C 01B9, 34C :3097 SRC/D.A,ALU/R.AND.S,DST/NOP,CIN/CIN
C 01BA, 34C :3098 SRC/D.A,ALU/R.AND.S,DST/NOP,CIN/CIN
C 01BB, 34C :3099 SRC/D.A,ALU/R.AND.S,DST/NOP,CIN/CIN
:3100 WR.XOR.LS:
C 01BC, 34E :3101 SRC/D.A,ALU/R.XOR.S,DST/NOP,CIN/CIN
C 01BD, 34E :3102 SRC/D.A,ALU/R.XOR.S,DST/NOP,CIN/CIN
C 01BE, 34E :3103 SRC/D.A,ALU/R.XOR.S,DST/NOP,CIN/CIN
C 01BF, 34E :3104 SRC/D.A,ALU/R.XOR.S,DST/NOP,CIN/CIN
:3105
:3106 Q.PLUS.LS:
C 01C0, 188 :3107 SRC/D.Q,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 01C1, 188 :3108 SRC/D.Q,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 01C2, 188 :3109 SRC/D.Q,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 01C3, 188 :3110 SRC/D.Q,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
:3111 LS.FROM.Q.LS:
C 01C4, 389 :3112 SRC/D.Q,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 01C5, 389 :3113 SRC/D.Q,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 01C6, 389 :3114 SRC/D.Q,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 01C7, 389 :3115 SRC/D.Q,ALU/S.MINUS.R,DST/NOP,CIN/CIN
:3116 Q.FROM.LS:
C 01C8, 38A :3117 SRC/D.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01C9, 38A :3118 SRC/D.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01CA, 38A :3119 SRC/D.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01CB, 38A :3120 SRC/D.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
:3121 Q.OR.LS:
C 01CC, 38B :3122 SRC/D.Q,ALU/R.OR.S,DST/NOP,CIN/CIN
C 01CD, 38B :3123 SRC/D.Q,ALU/R.OR.S,DST/NOP,CIN/CIN
```

ZZ-ENKCF-1.4
: ENKCF.MCR
: ENKCF.MIC

ENKCF.MIC

MICRO2 1M(01)

CONTROL ROM CONTENT

15-MAR-83 11:46:22

CONTROL ROM CONTENTS

M 6

Fiche 1 Frame M6

Sequence 77

ENKCF Micro-diagnostic for IDC module

Rev 01.40

Page 71

VER 00.01

72-

:

:

C 01CE, 38B	:3124	SRC/D.Q,ALU/R.OR.S,DST/NOP,CIN/CIN
C 01CF, 38B	:3125	SRC/D.Q,ALU/R.OR.S,DST/NOP,CIN/CIN
	:3126	
	:3127	Q.AND.LS:
C 01D0, 18C	:3128	SRC/D.Q,ALU/R.AND.S,DST/NOP,CIN/NO.CIN
C 01D1, 18C	:3129	SRC/D.Q,ALU/R.AND.S,DST/NOP,CIN/NO.CIN
C 01D2, 18C	:3130	SRC/D.Q,ALU/R.AND.S,DST/NOP,CIN/NO.CIN
C 01D3, 18C	:3131	SRC/D.Q,ALU/R.AND.S,DST/NOP,CIN/NO.CIN
	:3132	Q.XOR.LS:
C 01D4, 38E	:3133	SRC/D.Q,ALU/R.XOR.S,DST/NOP,CIN/CIN
C 01D5, 38E	:3134	SRC/D.Q,ALU/R.XOR.S,DST/NOP,CIN/CIN
C 01D6, 38E	:3135	SRC/D.Q,ALU/R.XOR.S,DST/NOP,CIN/CIN
C 01D7, 38E	:3136	SRC/D.Q,ALU/R.XOR.S,DST/NOP,CIN/CIN
	:3137	MOV.Q.LS:
C 01D8, 28B	:3138	SRC/O.Q,ALU/R.OR.S,DST/NOP,CIN/CIN
C 01D9, 28B	:3139	SRC/O.Q,ALU/R.OR.S,DST/NOP,CIN/CIN
C 01DA, 28B	:3140	SRC/O.Q,ALU/R.OR.S,DST/NOP,CIN/CIN
C 01DB, 28B	:3141	SRC/O.Q,ALU/R.OR.S,DST/NOP,CIN/CIN
	:3142	Q.NEG.LS:
C 01DC, 28A	:3143	SRC/O.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01DD, 28A	:3144	SRC/O.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01DE, 28A	:3145	SRC/O.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01DF, 28A	:3146	SRC/O.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
	:3147	
	:3148	WR.COM.LS:
C 01E0, 0CF	:3149	SRC/O.B,ALU/R.XNOR.S,DST/NOP,CIN/NO.CIN
C 01E1, 0CF	:3150	SRC/O.B,ALU/R.XNOR.S,DST/NOP,CIN/NO.CIN
C 01E2, 0CF	:3151	SRC/O.B,ALU/R.XNOR.S,DST/NOP,CIN/NO.CIN
C 01E3, 0CF	:3152	SRC/O.B,ALU/R.XNOR.S,DST/NOP,CIN/NO.CIN
	:3153	WR.NEG.LS:
C 01E4, 2CA	:3154	SRC/O.B,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01E5, 2CA	:3155	SRC/O.B,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01E6, 2CA	:3156	SRC/O.B,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01E7, 2CA	:3157	SRC/O.B,ALU/R.MINUS.S,DST/NOP,CIN/CIN
	:3158	Q.PLUS.WR.TO.LS:
C 01E8, 008	:3159	SRC/A.Q,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 01E9, 008	:3160	SRC/A.Q,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 01EA, 008	:3161	SRC/A.Q,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 01EB, 008	:3162	SRC/A.Q,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
	:3163	Q.FROM.WR.TO.LS:
C 01EC, 20A	:3164	SRC/A.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01ED, 20A	:3165	SRC/A.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01EE, 20A	:3166	SRC/A.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01EF, 20A	:3167	SRC/A.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
	:3168	
	:3169	DEC.LS:
C 01F0, 1CA	:3170	SRC/D.O,ALU/R.MINUS.S,DST/NOP,CIN/NO.CIN
C 01F1, 1CA	:3171	SRC/D.O,ALU/R.MINUS.S,DST/NOP,CIN/NO.CIN
C 01F2, 1CA	:3172	SRC/D.O,ALU/R.MINUS.S,DST/NOP,CIN/NO.CIN
C 01F3, 1CA	:3173	SRC/D.O,ALU/R.MINUS.S,DST/NOP,CIN/NO.CIN
	:3174	COM.LS:
C 01F4, 3CF	:3175	SRC/D.O,ALU/R.XNOR.S,DST/NOP,CIN/CIN
C 01F5, 3CF	:3176	SRC/D.O,ALU/R.XNOR.S,DST/NOP,CIN/CIN
C 01F6, 3CF	:3177	SRC/D.O,ALU/R.XNOR.S,DST/NOP,CIN/CIN
C 01F7, 3CF	:3178	SRC/D.O,ALU/R.XNOR.S,DST/NOP,CIN/CIN

ZZ-ENKCF-1.4
: ENKCF.MCR
: ENKCF.MIC

ENKCF.MIC

MICRO2 1M(01)

DIAGNOSTIC MACROS A15-MAR-83

15-MAR-83 11:46:22

ENKCF Micro-diagnostic for IDC module

Fiche 1 Frame B7

Sequence 79

Rev 01.40

Page 73

ZZ

```
:3191 .PAGE 'DIAGNOSTIC MACROS AND DEFINITIONS'
:3192
:3193 D.ADRS/=<15:9>,.VALIDITY=<D.VAL>,.DEFAULT=0
:3194
:3195
:3196 SAV.WR0=1 : STORAGE FOR WR0
:3197 SAV.WR1=2 : STORAGE FOR WR1
:3198 SAV.WR2=3 : STORAGE FOR WR2
:3199 SAV.WR3=4 : STORAGE FOR WR3
:3200 T5.SUB=5 : RESERVED FOR COMMON SUBROUTINES
:3201 T6.SUB=6 : RESERVED FOR COMMON SUBROUTINES
:3202 T7.SUB=7 : RESERVED FOR COMMON SUBROUTINES
:3203 TRANSFER.POINTER=7 : POINTER FOR TRANSFER ROUTINE
:3204 MEMORY.SIZE=7 : STORAGE FOR MEMORY SIZE IN 256KB BLOCKS AFTER
:3205 : SIZING
:3206 FPA.FOUND=7 : STORAGE FOR RESULT OF SIZING FOR FPA PRESENT
:3207 UBE.FOUND=7 : STORAGE FOR RESULT OF SIZING FOR UBE PRESENT
:3208 T8=8 : TEMP LOCATION 8
:3209 T9=9 : TEMP LOCATION 9
:3210 T10=0A : TEMP LOCATION 10
:3211 T11=0B : TEMP LOCATION 11
:3212 T12=0C : TEMP LOCATION 12
:3213 T13=0D : TEMP LOCATION 13
:3214 T14=0E : TEMP LOCATION 14
:3215 T15=0F : TEMP LOCATION 15
:3216 PC=10 : MACRO PROGRAM COUNTER
:3217
:3218 WR.BACKUP=11 : BACKUP WR FOR MOV MEM.DATA TO WR[]
:3219 MM.LASTADDR+1=12 : STORAGE OF LAST ADDRESS IN MAIN MEMORY PLUS
:3220 : 1 (FIRST NON-EXISTANT MEMORY ADDRESS)
:3221 #FF=13 : MASK
:3222 #FFFF=14 : MASK
:3223 #FF000000=15 : MASK
:3224 #FFFFFF00=16 : MASK
:3225 CSR0.MASK=16 : MASK
:3226 CSR1.MASK=17 : MASK (01003FFF)
:3227 CSR2.MASK=18 : MASK (7FFE3FFF)
:3228 #FE7FFFFFFF=19 : MASK
:3229 UBS.SET=1A : CONSTANT (7FF80000) USED BY UBS ADDRESS TEST
:3230 UBS.DATA=1B : MASK (7FFF8000) USED AS ERROR MASK FOR UBS
:3231 : DATA TEST
:3232
:3233 ERR.CON.NA=1E : CONSTANT OF 800500(H) BITS 23,10,8 SET FOR
:3234 : LOADING CONTROL AND STATUS REG IN INITIAL
:3235 : TESTS
:3236 ERR.CON=1F : CONSTANT OF 500(H) BITS 10 & 8 SET FOR
:3237 : LOADING CONTROL AND STATUS REG IN INITIAL
:3238 : TESTS
:3239
:3240 #1=20 : PATTERN 00000001 (HEX)
:3241 #2=21 : PATTERN 00000002
:3242 #4=22 : PATTERN 00000004
:3243 #8=23 : PATTERN 00000008
:3244 #10=24 : PATTERN 00000010
:3245 #20=25 : PATTERN 00000020
```


:3246	#40=26	: PATTERN 00000040
:3247	#80=27	: PATTERN 00000080
:3248	#100=28	: PATTERN 00000100
:3249	#200=29	: PATTERN 00000200
:3250	#400=2A	: PATTERN 00000400
:3251	#800=2B	: PATTERN 00000800
:3252	#1000=2C	: PATTERN 00001000
:3253	#2000=2D	: PATTERN 00002000
:3254	#4000=2E	: PATTERN 00004000
:3255	#8000=2F	: PATTERN 00008000
:3256	#10000=30	: PATTERN 00010000
:3257	#20000=31	: PATTERN 00020000
:3258	#40000=32	: PATTERN 00040000
:3259	#80000=33	: PATTERN 00080000
:3260	#100000=34	: PATTERN 00100000
:3261	#200000=35	: PATTERN 00200000
:3262	#400000=36	: PATTERN 00400000
:3263	#800000=37	: PATTERN 00800000
:3264	#1000000=38	: PATTERN 01000000
:3265	#2000000=39	: PATTERN 02000000
:3266	#4000000=3A	: PATTERN 04000000
:3267	#8000000=3B	: PATTERN 08000000
:3268	#10000000=3C	: PATTERN 10000000
:3269	#20000000=3D	: PATTERN 20000000
:3270	#40000000=3E	: PATTERN 40000000
:3271	#80000000=3F	: PATTERN 80000000
:3272		
:3273	BIT0=20	: PATTERN 00000001
:3274	BIT1=21	: PATTERN 00000002
:3275	BIT2=22	: PATTERN 00000004
:3276	BIT3=23	: PATTERN 00000008
:3277	BIT4=24	: PATTERN 00000010
:3278	BIT5=25	: PATTERN 00000020
:3279	BIT6=26	: PATTERN 00000040
:3280	BIT7=27	: PATTERN 00000080
:3281	BIT8=28	: PATTERN 00000100
:3282	BIT9=29	: PATTERN 00000200
:3283	BIT10=2A	: PATTERN 00000400
:3284	BIT11=2B	: PATTERN 00000800
:3285	BIT12=2C	: PATTERN 00001000
:3286	BIT13=2D	: PATTERN 00002000
:3287	BIT14=2E	: PATTERN 00004000
:3288	BIT15=2F	: PATTERN 00008000
:3289	BIT16=30	: PATTERN 00010000
:3290	BIT17=31	: PATTERN 00020000
:3291	BIT18=32	: PATTERN 00040000
:3292	BIT19=33	: PATTERN 00080000
:3293	BIT20=34	: PATTERN 00100000
:3294	BIT21=35	: PATTERN 00200000
:3295	BIT22=36	: PATTERN 00400000
:3296	BIT23=37	: PATTERN 00800000
:3297	BIT24=38	: PATTERN 01000000
:3298	BIT25=39	: PATTERN 02000000
:3299	BIT26=3A	: PATTERN 04000000
:3300	BIT27=3B	: PATTERN 08000000

```

:3301      BIT28=3C      ; PATTERN 10000000
:3302      BIT29=3D      ; PATTERN 20000000
:3303      BIT30=3E      ; PATTERN 40000000
:3304      BIT31=3F      ; PATTERN 80000000
:3305
:3306
:3307      ;CSR ADDRESS MNEMONICS
:3308
:3309      CSR0=4E      ; ALL BITS CLEAR TO ACCESS CSR 0
:3310      CSR1=22      ; BIT 2 SET TO ACCESS CSR 1
:3311      CSR2=23      ; BIT 3 SET TO ACCESS CSR 2
:3312
:3313      ;CONTROL AND STATUS WORD BITS
:3314
:3315      PRINT.UBE=26  ; PRINT IF UBE PRESENT OR NOT PRESENT (BIT 6)
:3316                        ; (INDICATOR IN LS 7)
:3317      PRINT.R80=27  ; PRINT IF R80 PRESENT OR NOT PRESENT AND ON
:3318                        ; WHICH DRIVE (BIT 7). INDICATOR IN LS 7.
:3319                        ; DRIVE NUMBER IN LS 6.
:3320      ERROR=28      ; TEST ERROR INDICATION (BIT 8)
:3321      SET.PA.ERR=29 ; TELLS CONSOLE TO CREATE A PARITY ERROR AT WCS
:3322                        ; ADDRESS POINTED TO BY LS 7 (BIT 9)
:3323      CONSOLE.LOE=2A ; INDICATES CONSOLE DOES ERROR LOOPING (BIT 10)
:3324                        ; (FOR INITIAL TESTS)
:3325      PRINT.MEM.SIZE=2B ; PRINT MEMORY SIZE IN 256K BLOCKS (BIT 11)
:3326                        ; (SIZE IN LS 7)
:3327      PRINT.FPA=2C   ; PRINT IF FPA PRESENT OR NOT PRESENT (BIT 12)
:3328                        ; (INDICATOR IN LS 7)
:3329      XFER.DATA=2D   ; INDICATES A DATA TRANSFER TO LS OR MM (BIT 13)
:3330      EOS=2E         ; END OF SECTION (BIT 14)
:3331      BEGIN.TEST=2F  ; BEGINNING OF TEST (BIT 15)
:3332      INTERRUPT.EN=30 ; ENABLE INTERRUPT OR SIGNAL SPECIFIED BY BITS
:3333                        ; 17 THRU 22 AND 28 THRU 30 (BIT 16)
:3334
:3335      CONSOLE.HALT=31 ; CONSOLE HALT INTERRUPT (BIT 17)
:3336      PWR.FAIL=32     ; PWR FAIL INT (BIT 18)
:3337      INTERVAL.TIM=33 ; INTRVL TIM INT (BIT 19)
:3338      CONSOLE.ATTN=34 ; CONSOLE ATTN INTERRUPT (BIT 20)
:3339      CONSOLE.ACK=35  ; CONSOLE ACK INTERRUPT (BIT 21)
:3340      DISABLE.MEM.REF=36 ; DISABLE MEMORY REFERENCES (BIT 22)
:3341      CINIT=3C       ; UNIBUS INIT SIGNAL TO THE MCT (BIT 28)
:3342      UBS.DCLO=3D     ; UNIBUS DC LO INDICATION TO THE MCT (BIT 29)
:3343      UBS.BBSY=3E     ; UNIBUS BUS BUSY INDICATOR TO MCT (BIT 30)
:3344
:3345      NA.EXP.REC=37    ; PRINT N/A UNDER EXPECTED AND RECEIVED (BIT 23)
:3346      OTHER.DATA=38   ; PRINT A VALUE UNDER OTHER DATA (BIT 24)
:3347      CONSOLE.LOOP=39 ; CONSOLE PERFORMS A LOOP ACCORDING TO LOOP
:3348                        ; COUNT IN LS (BIT 25)
:3349      PRINT.CRD=3A     ; PRINT NUMBER OF SINGLE BIT ERRORS (BIT 26)
:3350      CLR.PA.ERR=3B    ; TELLS CONSOLE TO CLEAR PARITY ERROR AT WCS
:3351                        ; ADDRESS POINTED TO BY LS 7 (BIT 27)
:3352      PRINT.IDC=3F     ; PRINT IF IDC PRESENT OR NOT PRESENT (BIT 31)
:3353                        ; (INDICATOR IN LS 7)
:3354
:3355      ;MODULE CODES
  
```

```

:3356
:3357      WCS=20      ; MODULE CODE FOR WCS BOARD (BIT 0 SET)
:3358      CPU=21     ; MODULE CODE FOR CPU BOARD (BIT 1 SET)
:3359      MCT=22     ; MODULE CODE FOR MCT BOARD (BIT 2 SET)
:3360      FPA=23     ; MODULE CODE FOR FPA BOARD (BIT 3 SET)
:3361      ARRAY=24   ; MODULE CODE FOR ARRAY BOARD (BIT 4 SET)
:3362      IDC=25     ; MODULE CODE FOR IDC BOARD (BIT 5 SET)
:3363
:3364      ;CSR 1 ERROR BITS
:3365
:3366      VALID.ERR=2E ; VALID NOT SET IF 1 (BIT 14)
:3367      TB.PAR.ERR=2F ; TB PARITY ERROR (BIT 15)
:3368      NXM=30      ; NON-EXISTANT MEMORY ERROR (BIT 16)
:3369      UB.BSY=31   ; UNIBUS BUSY (BIT 17)
:3370      ADP.REG=32   ; UNIBUS ADAPTER REGISTER SELECT (BIT 18)
:3371      WR.ACROSS.PG=33 ; WRITE ACROSS PAGE BOUNDARY ERROR (BIT 19)
:3372      OP.ERR=34   ; OPERATION ERROR (BIT 20)
:3373      TB.MISS=35  ; TB MISS (VIRTUAL ADDRESS TRANSLATION NOT IN
:3374                      ; BUFFER) (BIT 21)
:3375      ACCESS.REFUSED=36 ; PROTECTION ERROR ON ACCESS (BIT 22)
:3376      MODIFY.REFUSED=37 ; PROTECTION ERROR ON WRITE (BIT 23)
:3377
:3378      CRD=3E      ; SINGLE BIT ERROR CORRECTED (BIT 30)
:3379      RDS=3F      ; UNCORRECTABLE DATA ERROR (BIT 31)
:3380
:3381
:3382      ;CSR 1 CONTROL BITS
:3383
:3384      ECC.DIS=39   ; CSR1 ECC DISABLE BIT (BIT 25)
:3385      DIAG.CHK=3A  ; CSR1 DIAG CHK BIT (BIT 26)
:3386      MME=3B      ; CSR1 MEMORY MANAGMENT ENABLE BIT (BIT 27)
:3387      INH.CRD=3C  ; CSR1 INHIBIT REPORT CRD BIT (BIT 28)
:3388      TB.PAR.DIAG=3D ; CSR1 TB PAR DIAG BIT (FORCE TB PARITY ERR)
:3389                      ; (BIT 29)
:3390
:3391
:3392      ;UBE CONTROL REGISTER BITS
:3393
:3394      ;CONTROL REG 1
:3395      GO=20       ; START UBE (BIT 0)
:3396      BR4=21      ; ISSUE BUS REQUEST 4 (BIT 1)
:3397      BR5=22      ; ISSUE BUS REQUEST 5 (BIT 2)
:3398      BR6=23      ; ISSUE BUS REQUEST 6 (BIT 3)
:3399      BR7=24      ; ISSUE BUS REQUEST 7 (BIT 4)
:3400      NPR=25      ; ISSUE NPR (BIT 5)
:3401      INT.ODNE=26 ; INTERRUPT ON DONE (WHEN CCOVF) (BIT 6)
:3402      RDY=27      ; READY BIT, SET WHEN READY TO BEGIN FUNC (BIT 7)
:3403      C0=28       ; C0 BIT (BIT 8)
:3404      C1=29       ; C1 BIT (BIT 9)
:3405      FUNA=2A      ; FUNCTION BIT A (BIT 10)
:3406      FUNB=2B      ; FUNCTION BIT B (BIT 11)
:3407      CC+DAT=2C    ; DATA FROM CYCLE COUNT IF SET, DATA REG IF CLEAR
:3408      INH.DATIP.ROL=2D ; INHIBIT ROL ON DATIP (BIT 13)
:3409      DATOB.ON.DATIP=2E ; DO DATOB AFTER DATIP CYCLE (BIT 14)
:3410      ERR=2F      ; EXERCISOR OR UNIBUS ERROR (BIT 15)

```

```

:3411
:3412 ;CONTROL REG 2
:3413 EXT.MEM0=20 ; EXTENDED MEMORY BIT 0 (BIT 0)
:3414 EXT.MEM1=21 ; EXTENDED MEMORY BIT 1 (BIT 1)
:3415 INH.INC.ADDR&CC=22 ; INHIBIT INC ON CYCLE COUNT AND ADDR REG (BIT 2)
:3416 INH.SACK=23 ; INHIBIT ASSERTING BUS SACK ON BUS GRANT (BIT 3)
:3417 PWR.DWN.SEQ=24 ; ASSERT ACLO (BIT 4)
:3418 WNG.GNT.BCK=25 ; NO BUS GRANT RECV TO HIGHEST REQUEST (BIT 5)
:3419 MAX.LATE.ERR=26 ; NPR AND BR LATENCY TIME EXCEEDED (BIT 6)
:3420 NO.SACK.ERR=27 ; NO TIMEOUT AT CPU WHEN NO SACK (BIT 7)
:3421 NO.SSYN.ERR=28 ; NO SSYN RECEIVED AFTER ASSERTING MSYN (BIT 8)
:3422 WRG.A.LINES=29 ; ADDRESS ERR ON SENT & RECEIVED ADDRESS (BIT 9)
:3423 NO.GNT+NOT.ONE.GNT=2A ; NO GRANT OR MORE THAN 1 GRANT AFTER REQ. (BIT 10)
:3424 INTR.SSYN.ERR=2B ; NO SSYN RECEIVED AFTER BUS INTR (BIT 11)
:3425 ENB.PB=2C ; ENABLE PARITY BIT B (BIT 12)
:3426 CCOVF=2D ; CYCLE COUNT OVERFLOW (BIT 13)
:3427 TM.DLY=2E ; REQUEST BUS EVERY 10 USEC (BIT 14)
:3428
:3429 ;BG6 MNEMONIC
:3430 BG6=23 ; BIT 3 SET FOR BG 6 ADDRESS
:3431
:3432 ;ERROR AND CONTROL INFORMATION
:3433
:3434 CONTROL.STATUS=40 ; CONTROL AND STATUS WORD
:3435 ERROR.NUMBER=41 ; ERROR NUMBER OF CURRENT TEST
:3436 EXPECTED.DATA=42 ; EXPECTED RESULT OF CURRENT TEST
:3437 RECEIVED.DATA=43 ; ACTUAL RESULT OF CURRENT TEST
:3438 ADDRESS.DATA=44 ; OTHER PERTINENT DATA
:3439 ERROR.MASK=45 ; BITS TO CHECK IN RESULT
:3440 MODULE.NUM=46 ; STORAGE OF MODULE NUMBER (CODED)
:3441 LOOP.CNT=47 ; STORAGE OF LOOP COUNT FOR CONSOLE LOOP
:3442 CONTROL.
:3443 ERROR.CONTROL=48 ; STORAGE FOR ERROR CONTROL (LOOP ON ERROR ETC.)
:3444 LOE=20 ; LOOP ON ERROR IF SET (BIT0)
:3445 NER=21 ; NO ERROR REPORT IF SET (BIT1)
:3446 BELL=22 ; BELL ON ERROR IF SET (BIT2)
:3447 HOE=23 ; HALT ON ERROR IF SET (BIT3)
:3448 SER=24 ; ENABLE ERROR REPORT ON CRD ERRORS IF SET
:3449 APT.PRESENT=25 ; APT PRESENT IF SET (BIT5)
:3450 LOOP.COMMAND=26 ; LOOP COMMAND TYPED IF SET (BIT 6)
:3451 SPECIAL=27 ; SPECIAL BIT FOR LOOP ON SIZE TESTS (BIT 7)
:3452
:3453 APT.PARAM=49 ; APT RUN TIME PARAMETER REGS (MEM SIZE, HARD-
:3454 ; WARE CONFIG, SOFTWARE CONFIG IN BYTES 0, 1
:3455 ; AND 2 RESPECTIVELY. BYTE 3 FOR FUTURE USE)
:3456 APT.RESERVED=4A ; RESERVED FOR FUTURE APT USE
:3457
:3458 ;MISCELLANEOUS CONSTANTS AND STORAGE
:3459
:3460 #AAAAAAA=4C ; CONSTANT
:3461 ALTER.ODD=4C ; CONSTANT
:3462 #5555555=4D ; CONSTANT
:3463 ALTER.EVEN=4D ; CONSTANT
:3464 #0=4E ; CONSTANT
:3465 ZERO=4E ; CONSTANT

```

```

:3466      #FFFFFFF=4F      ; CONSTANT
:3467      ONES=4F          ; CONSTANT
:3468      PREVIOUS.ERROR=50 ; ERROR NUMBER OF LAST ERROR
:3469
:3470      DATA.1=51        ; STORAGE FOR FPA DATA
:3471      DATA.2=52        ; STORAGE FOR FPA DATA
:3472      DATA.3=53        ; STORAGE FOR FPA DATA
:3473      FIRST.FLT=54      ; STORAGE FOR FPA DATA
:3474      SEC.FLT=55        ; STORAGE FOR FPA DATA
:3475      THIRD.FLT=56      ; STORAGE FOR FPA DATA
:3476      T57=57           ; TEMP LOCATION 57
:3477      T58=58           ; TEMP LOCATION 58
:3478      T59=59           ; TEMP LOCATION 59
:3479
:3480      BEDB=5A          ;UBE (BUS EXERCISOR) DATA BUF REG
:3481      BECC=5B          ;UBE (BUS EXERCISOR) CYCLE COUNT REG
:3482      BEBA=5C          ;UBE (BUS EXERCISOR) ADDR REG
:3483      BECR1=5D         ;UBE (BUS EXERCISOR) CONTROL REG 1
:3484      CLEAR.ERR.ADDR=5E ;UBE CLEAR ERROR REG ADDRESS
:3485      BECR2=5F         ;UBE (BUS EXERCISOR) CONTROL REG 2
:3486
:3487      #3(H)=60         ; CONSTANT
:3488      #12(H)=61        ; CONSTANT
:3489      #33333333=62     ; CONSTANT
:3490      #0F0F0F0F=63     ; CONSTANT
:3491      #00FF00FF=64     ; CONSTANT
:3492      #162(H)=65       ; CONSTANT FOR LS TRANSFER POINTER
:3493      BR7.IDENT=66     ; BR7 IDENTIFIER (1010(B) IN BITS 5-2)
:3494      BG7=67          ; BG7 ADDRESS (BITS 2 AND 3 SET)
:3495
:3496      ;TEMPS FOR CPU TESTS
:3497      L30=30           ;LS 30 TEMP (RESTORED TO 10000 AFTER USE)
:3498      L31=31           ;LS 31 TEMP (RESTORE TO 20000 AFTER USE)
:3499      L53=53           ;LS 53 TEMP
:3500      L73=73           ;LS 73 TEMP
:3501
:3502      ;REGISTER ADDRESSES
:3503
:3504      CONTROL.OS=78      ; HARDWARE INDEX TO CONTROL WORDS (LS 40-4F)
:3505      SHIFT.OS(3-0)=79 ; 16 LOCATION HARDWARE INDEX TO SHIFT PATTERN
:3506      ; (LS 20-2F)
:3507      SHIFT.OS(4-0)=7B ; 32 LOCATION HARDWARE INDEX TO SHIFT PATTERN
:3508      ; (LS 20-3F)
:3509      OS=7C             ; OS REGISTER
:3510      DEC.CON=7D        ; DECIMAL CARRIES
:3511      SIZE=7E           ; SIZE REGISTER
:3512      MDT= 7E           ; MEMORY REFERENCE DATA SIZE
:3513      INTERRUPT.VEC=7E  ; HARDWARE INTERRUPT VECTOR
:3514      ;
:3515      ; THIS WORD IS THE HARDWARE CONDITION CODES. THE CC'S CAN BE READ
:3516      ; OR ARE WRITTEN HERE. NO OTHER BITS IN THE PSL ARE EFFECTED.
:3517      ;
:3518      PSL.CC=7F         ; PSL CONDITION CODES
    
```

```

:3519 .PAGE
:3520 XD.ADRS/= <16:9> ,.VALIDITY=<XD.VAL>
:3521
:3522
:3523 SAV.WR0=1 ; STORAGE FOR WR0
:3524 SAV.WR1=2 ; STORAGE FOR WR1
:3525 SAV.WR2=3 ; STORAGE FOR WR2
:3526 SAV.WR3=4 ; STORAGE FOR WR3
:3527 T5.SUB=5 ; RESERVED FOR COMMON SUBROUTINES
:3528 T6.SUB=6 ; RESERVED FOR COMMON SUBROUTINES
:3529 T7.SUB=7 ; RESERVED FOR COMMON SUBROUTINES
:3530 TRANSFER.POINTER=7 ; POINTER FOR TRANSFER ROUTINE
:3531 MEMORY.SIZE=7 ; STORAGE FOR MEMORY SIZE IN 256KB BLOCKS AFTER
:3532 ; SIZING
:3533 FPA.FOUND=7 ; STORAGE FOR RESULT OF SIZING FOR FPA PRESENT
:3534 UBE.FOUND=7 ; STORAGE FOR RESULT OF SIZING FOR UBE PRESENT
:3535 T8=8 ; TEMP LOCATION 8
:3536 T9=9 ; TEMP LOCATION 9
:3537 T10=0A ; TEMP LOCATION 10
:3538 T11=0B ; TEMP LOCATION 11
:3539 T12=0C ; TEMP LOCATION 12
:3540 T13=0D ; TEMP LOCATION 13
:3541 T14=0E ; TEMP LOCATION 14
:3542 T15=0F ; TEMP LOCATION 15
:3543 PC=10 ; MACRO PROGRAM COUNTER
:3544
:3545 WR.BACKUP=11 ; BACKUP WR FOR MOV MEM.DATA TO WR[]
:3546 MM.LASTADDR+1=12 ; STORAGE OF LAST ADDRESS IN MAIN MEMORY PLUS
:3547 ; 1 (FIRST NON-EXISTANT MEMORY ADDRESS)
:3548 #FF=13 ; MASK
:3549 #FFFF=14 ; MASK
:3550 #FF000000=15 ; MASK
:3551 #FFFFFF00=16 ; MASK
:3552 CSR0.MASK=16 ; MASK
:3553 CSR1.MASK=17 ; MASK (01003FFF)
:3554 CSR2.MASK=18 ; MASK (FFFE3FFF)
:3555 #FE7FFFFFFF=19 ; MASK
:3556 UBS.SET=1A ; CONSTANT (7FF80000) USED BY UBS ADDRESS TEST
:3557 UBS.DATA=1B ; MASK (7FFF8000) USED AS ERROR MASK FOR UBS
:3558 ; DATA TEST
:3559
:3560 ERR.CON.NA=1E ; CONSTANT OF 800500(H) BITS 23,10,8 SET FOR
:3561 ; LOADING CONTROL AND STATUS REG IN INITIAL
:3562 ; TESTS
:3563 ERR.CON=1F ; CONSTANT OF 500(H) BITS 10 & 8 SET FOR
:3564 ; LOADING CONTROL AND STATUS REG IN INITIAL
:3565 ; TESTS
:3566
:3567 #1=20 ; PATTERN 00000001 (HEX)
:3568 #2=21 ; PATTERN 00000002
:3569 #4=22 ; PATTERN 00000004
:3570 #8=23 ; PATTERN 00000008
:3571 #10=24 ; PATTERN 00000010
:3572 #20=25 ; PATTERN 00000020
:3573 #40=26 ; PATTERN 00000040

```

:3574	#80=27	: PATTERN 00000080
:3575	#100=28	: PATTERN 00000100
:3576	#200=29	: PATTERN 00000200
:3577	#400=2A	: PATTERN 00000400
:3578	#800=2B	: PATTERN 00000800
:3579	#1000=2C	: PATTERN 00001000
:3580	#2000=2D	: PATTERN 00002000
:3581	#4000=2E	: PATTERN 00004000
:3582	#8000=2F	: PATTERN 00008000
:3583	#10000=30	: PATTERN 00010000
:3584	#20000=31	: PATTERN 00020000
:3585	#40000=32	: PATTERN 00040000
:3586	#80000=33	: PATTERN 00080000
:3587	#100000=34	: PATTERN 00100000
:3588	#200000=35	: PATTERN 00200000
:3589	#400000=36	: PATTERN 00400000
:3590	#800000=37	: PATTERN 00800000
:3591	#1000000=38	: PATTERN 01000000
:3592	#2000000=39	: PATTERN 02000000
:3593	#4000000=3A	: PATTERN 04000000
:3594	#8000000=3B	: PATTERN 08000000
:3595	#10000000=3C	: PATTERN 10000000
:3596	#20000000=3D	: PATTERN 20000000
:3597	#40000000=3E	: PATTERN 40000000
:3598	#80000000=3F	: PATTERN 80000000
:3599		
:3600	BIT0=20	: PATTERN 00000001
:3601	BIT1=21	: PATTERN 00000002
:3602	BIT2=22	: PATTERN 00000004
:3603	BIT3=23	: PATTERN 00000008
:3604	BIT4=24	: PATTERN 00000010
:3605	BIT5=25	: PATTERN 00000020
:3606	BIT6=26	: PATTERN 00000040
:3607	BIT7=27	: PATTERN 00000080
:3608	BIT8=28	: PATTERN 00000100
:3609	BIT9=29	: PATTERN 00000200
:3610	BIT10=2A	: PATTERN 00000400
:3611	BIT11=2B	: PATTERN 00000800
:3612	BIT12=2C	: PATTERN 00001000
:3613	BIT13=2D	: PATTERN 00002000
:3614	BIT14=2E	: PATTERN 00004000
:3615	BIT15=2F	: PATTERN 00008000
:3616	BIT16=30	: PATTERN 00010000
:3617	BIT17=31	: PATTERN 00020000
:3618	BIT18=32	: PATTERN 00040000
:3619	BIT19=33	: PATTERN 00080000
:3620	BIT20=34	: PATTERN 00100000
:3621	BIT21=35	: PATTERN 00200000
:3622	BIT22=36	: PATTERN 00400000
:3623	BIT23=37	: PATTERN 00800000
:3624	BIT24=38	: PATTERN 01000000
:3625	BIT25=39	: PATTERN 02000000
:3626	BIT26=3A	: PATTERN 04000000
:3627	BIT27=3B	: PATTERN 08000000
:3628	BIT28=3C	: PATTERN 10000000

[illegible]


```

:3684 ;MODULE CODES
:3685
:3686     WCS=20      ; MODULE CODE FOR WCS BOARD (BIT 0 SET)
:3687     CPU=21     ; MODULE CODE FOR CPU BOARD (BIT 1 SET)
:3688     MCT=22     ; MODULE CODE FOR MCT BOARD (BIT 2 SET)
:3689     FPA=23     ; MODULE CODE FOR FPA BOARD (BIT 3 SET)
:3690     ARRAY=24   ; MODULE CODE FOR ARRAY BOARD (BIT 4 SET)
:3691     IDC=25     ; MODULE CODE FOR IDC BOARD (BIT 5 SET)
:3692
:3693 ;CSR 1 ERROR BITS
:3694
:3695     VALID.ERR=2E ; VALID NOT SET IF 1 (BIT 14)
:3696     TB.PAR.ERR=2F ; TB PARITY ERROR (BIT 15)
:3697     NXM=30       ; NON-EXISTANT MEMORY ERROR (BIT 16)
:3698     UB.BSY=31    ; UNIBUS BUSY (BIT 17)
:3699     ADP.REG=32   ; UNIBUS ADAPTER REGISTER SELECT (BIT 18)
:3700     WR.ACROSS.PG=33 ; WRITE ACROSS PAGE BOUNDARY ERROR (BIT 19)
:3701     OP.ERR=34   ; OPERATION ERROR (BIT 20)
:3702     TB.MISS=35  ; TB MISS (VIRTUAL ADDRESS TRANSLATION NOT IN
:3703                 ; BUFFER) (BIT 21)
:3704     ACCESS.REFUSED=36 ; PROTECTION ERROR ON ACCESS (BIT 22)
:3705     MODIFY.REFUSED=37 ; PROTECTION ERROR ON WRITE (BIT 23)
:3706
:3707     CRD=3E        ; SINGLE BIT ERROR CORRECTED (BIT 30)
:3708     RDS=3F        ; UNCORRECTABLE DATA ERROR (BIT 31)
:3709
:3710 ;CSR 1 CONTROL BITS
:3711
:3712     ECC.DIS=39    ; CSR1 ECC DISABLE BIT (BIT 25)
:3713     DIAG.CHK=3A  ; CSR1 DIAG CHK BIT (BIT 26)
:3714     MME=3B       ; CSR1 MEMORY MANAGMENT ENABLE BIT (BIT 27)
:3715     INH.CRD=3C   ; CSR1 INHIBIT REPORT CRD BIT (BIT 28)
:3716     TB.PAR.DIAG=3D ; CSR1 TB PAR DIAG BIT (FORCE TB PARITY ERR)
:3717                 ; (BIT 29)
:3718
:3719
:3720 ;UBE CONTROL REGISTER BITS
:3721
:3722 ;CONTROL REG 1
:3723
:3724     GO=20         ; START UBE (BIT 0)
:3725     BR4=21        ; ISSUE BUS REQUEST 4 (BIT 1)
:3726     BR5=22        ; ISSUE BUS REQUEST 5 (BIT 2)
:3727     BR6=23        ; ISSUE BUS REQUEST 6 (BIT 3)
:3728     BR7=24        ; ISSUE BUS REQUEST 7 (BIT 4)
:3729     NPR=25        ; ISSUE NPR (BIT 5)
:3730     INT.ODNE=26   ; INTERRUPT ON DONE (WHEN CCOVF) (BIT 6)
:3731     RDY=27        ; READY BIT, SET WHEN READY TO BEGIN FUNC (BIT 7)
:3732     C0=28         ; C0 BIT (BIT 8)
:3733     C1=29         ; C1 BIT (BIT 9)
:3734     FUNA=2A       ; FUNCTION BIT A (BIT 10)
:3735     FUNB=2B       ; FUNCTION BIT B (BIT 11)
:3736     CC+DAT=2C      ; DATA FROM CYCLE COUNT IF SET, DATA REG IF CLEAR
:3737     INH.DATIP.ROL=2D ; INHIBIT ROL ON DATIP (BIT 13)
:3738     DATOB.ON.DATIP=2E ; DO DATOB AFTER DATIP CYCLE (BIT 14)

```

ZZ-ENKCF-1.4
: ENKCF.MCR
: ENKCF.MIC

ENKCF.MIC

MICRO2 1M(01)

DIAGNOSTIC MACROS A15-MAR-83

15-MAR-83 11:46:22

DIAGNOSTIC MACROS AND DEFINITIONS

Fiche 1 Frame L7

Sequence 89

Page 83

ZZ-1

```
:3739      ERR=2F      ; EXERCISOR OR UNIBUS ERROR (BIT 15)
:3740
:3741      ;CONTROL REG 2
:3742      EXT.MEM0=20      ; EXTENDED MEMORY BIT 0 (BIT 0)
:3743      EXT.MEM1=21      ; EXTENDED MEMORY BIT 1 (BIT 1)
:3744      INH.INC.ADDR&CC=22      ; INHIBIT INC ON CYCLE COUNT AND ADDR REG (BIT 2)
:3745      INH.SACK=23      ; INHIBIT ASSERTING BUS SACK ON BUS GRANT (BIT 3)
:3746      PWR.DWN.SEQ=24      ; ASSERT ACLO (BIT 4)
:3747      WNG.GNT.BCK=25      ; NO BU. GRANT RECV TO HIGEST REQUEST (BIT 5)
:3748      MAX.LATE.ERR=26      ; NPR AND BR LATENCY TIME EXCEEDED (BIT 6)
:3749      NO.SACK.ERR=27      ; NO TIMEOUT AT CPU WHEN NO SACK (BIT 7)
:3750      NO.SSYN.ERR=28      ; NO SSYN RECEIVED AFTER ASSERTING MSYN (BIT 8)
:3751      WRG.A.LINES=29      ; ADDRESS ERR ON SENT & RECEIVED ADDRESS (BIT 9)
:3752      NO.GNT+NOT.ONE.GNT=2A      ; NO GRANT OR MORE THAN 1 GRANT AFTER REQ. (BIT 10)
:3753      INTR.SSYN.ERR=28      ; NO SSYN RECEIVED AFTER BUS INTR (BIT 11)
:3754      ENB.PB=2C      ; ENABLE PARITY BIT B (BIT 12)
:3755      CCOVF=2D      ; CYCLE COUNT OVERFLOW (BIT 13)
:3756      TM.DLY=2E      ; REQUEST BUS EVERY 10 USEC (BIT 14)
:3757
:3758      ;BG6 MNEMONIC
:3759      BG6=23      ; BIT 3 SET FOR BG 6 ADDRESS
:3760
:3761      ;ERROR AND CONTROL INFORMATION
:3762
:3763      CONTROL.STATUS=40      ; CONTROL AND STATUS WORD
:3764      ERROR.NUMBER=41      ; ERROR NUMBER OF CURRENT TEST
:3765      EXPECTED.DATA=42      ; EXPECTED RESULT OF CURRENT TEST
:3766      RECEIVED.DATA=43      ; ACTUAL RESULT OF CURRENT TEST
:3767      ADDRESS.DATA=44      ; OTHER PERTINENT DATA
:3768      ERROR.MASK=45      ; BITS TO CHECK IN RESULT
:3769      MODULE.NUM=46      ; STORAGE OF MODULE NUMBER (CODED)
:3770      LOOP.CNT=47      ; STORAGE OF LOOP COUNT FOR CONSOLE LOOP
:3771      ; CONTROL
:3772      ERROR.CONTROL=48      ; STORAGE FOR ERROR CONTROL (LOOP ON ERROR ETC.)
:3773      LOE=20      ; LOOP ON ERROR IF SET (BIT0)
:3774      NER=21      ; NO ERROR REPORT IF SET (BIT1)
:3775      BELL=22      ; BELL ON ERROR IF SET (BIT2)
:3776      HOE=23      ; HALT ON ERROR IF SET (BIT3)
:3777      SER=24      ; ENABLE ERROR REPORT ON CRD ERRORS IF SET
:3778      APT.PRESENT=25      ; APT PRESENT IF SET (BIT5)
:3779      LOOP.COMMAND=26      ; LOOP COMMAND TYPED IF SET (BIT 6)
:3780      SPECIAL=27      ; SPECIAL BIT FOR LOOP ON SIZE TESTS (BIT 7)
:3781
:3782      APT.PARAM=49      ; APT RUN TIME PARAMETER REGS (MEM SIZE, HARD-
:3783      ; WARE CONFIG, SOFTWARE CONFIG IN BYTES 0, 1
:3784      ; AND 2 RESPECTIVELY. BYTE 3 FOR FUTURE USE)
:3785      APT.RESERVED=4A      ; RESERVED FOR FUTURE APT USE
:3786
:3787      ;MISCELLANEOUS CONSTANTS AND STORAGE
:3788
:3789      #AAAAAAA=4C      ; CONSTANT
:3790      ALTER.ODD=4C      ; CONSTANT
:3791      #55555555=4D      ; CONSTANT
:3792      ALTER.EVEN=4D      ; CONSTANT
:3793      #0=4E      ; CONSTANT
```

72-ENKCF-1.4
: ENKCF.MCR
: ENKCF.MIC

ENKCF.MIC

MICRO2 1M(01)

DIAGNOSTIC MACROS A15-MAR-83

15-MAR-83 11:46:22

ENKCF Micro-diagnostic for IDC module

Sequence 90

Page 84

72-

:

:

```
:3794      ZERO=4E      ; CONSTANT
:3795      #FFFFFFFF=4F ; CONSTANT
:3796      ONES=4F      ; CONSTANT
:3797      PREVIOUS.ERROR=50 ; ERROR NUMBER OF LAST ERROR
:3798
:3799      DATA.1=51    ; STORAGE FOR FPA DATA
:3800      DATA.2=52    ; STORAGE FOR FPA DATA
:3801      DATA.3=53    ; STORAGE FOR FPA DATA
:3802      FIRST.FLT=54  ; STORAGE FOR FPA DATA
:3803      SEC.FLT=55    ; STORAGE FOR FPA DATA
:3804      THIRD.FLT=56  ; STORAGE FOR FPA DATA
:3805      T57=57        ; TEMP LOCATION 57
:3806      T58=58        ; TEMP LOCATION 58
:3807      T59=59        ; TEMP LOCATION 59
:3808
:3809      BEDB=5A        ;UBE (BUS EXERCISOR) DATA BUF REG
:3810      BECC=5B        ;UBE (BUS EXERCISOR) CYCLE COUNT REG
:3811      BEBA=5C        ;UBE (BUS EXERCISOR) ADDR REG
:3812      BECR1=5D       ;UBE (BUS EXERCISOR) CONTROL REG 1
:3813      CLEAR.ERR.ADDR=5E ;UBE CLEAR ERROR REG ADDRESS
:3814      BECR2=5F       ;UBE (BUS EXERCISOR) CONTROL REG 2
:3815
:3816      #3(H)=60      ; CONSTANT
:3817      #12(H)=61     ; CONSTANT
:3818      #33333333=62  ; CONSTANT
:3819      #0F0F0F0F=63  ; CONSTANT
:3820      #00FF00FF=64  ; CONSTANT
:3821      #162(H)=65    ; CONSTANT FOR LS TRANSFER POINTER
:3822      BR7.IDENT=66  ; BR7 IDENTIFIER (1010(B) IN BITS 5-2)
:3823      BG7=67        ; BG7 ADDRESS (BITS 2 AND 3 SET)
:3824
:3825      ;TEMPS FOR CPU TESTS
:3826      L30=30        ;LS 30 TEMP (RESTORED TO 10000 AFTER USE)
:3827      L31=31        ;LS 31 TEMP (RESTORE TO 20000 AFTER USE)
:3828      L53=53        ;LS 53 TEMP
:3829      L73=73        ;LS 73 TEMP
:3830
:3831      ;REGISTER ADDRESSES
:3832
:3833      CONTROL.OS=78   ; HARDWARE INDEX TO CONTROL WORDS (LS 40-4F)
:3834      SHIFT.OS(3-0)=79 ; 16 LOCATION HARDWARE INDEX TO SHIFT PATTERN
:3835      ; (LS 20-2F)
:3836      SHIFT.OS(4-0)=7B ; 32 LOCATION HARDWARE INDEX TO SHIFT PATTERN
:3837      ; (LS 20-3F)
:3838      OS=7C           ; OS REGISTER
:3839      DEC.CON=7D      ; DECIMAL CARRIES
:3840      SIZE=7E         ; SIZE REGISTER
:3841      MDT= 7E         ; MEMORY REFERENCE DATA SIZE
:3842      INTERRUPT.VEC=7E ; HARDWARE INTERRUPT VECTOR
:3843      ;
:3844      ; THIS WORD IS THE HARDWARE CONDITION CODES. THE CC'S CAN BE READ
:3845      ; OR ARE WRITTEN HERE. NO OTHER BITS IN THE PSL ARE EFFECTED.
:3846      ;
:3847      PSL.CC=7F       ; PSL CONDITION CODES
:3848
```

```

:3849
:3850 .TOC    "COMMON LS ASSIGNMENTS (TO REGISTERS)"
:3851
:3852 .UPPER HALF OF LS
:3853
:3854     XGPR.OS=0F8           ; HARDWARE INDEX TO XGPRS
:3855     USER.INDEX(3-0)=0F9 ; 16 LOCATION HARDWARE INDEX TO DIAGNOSTIC DATA
:3856                               ; AREA (LS LOCATIONS 0A0 THROUGH 0AF)
:3857     USER.INDEX(4-0)=0FB ; 32 LOCATION HARDWARE INDEX TO DIAGNOSTIC DATA
:3858                               ; AREA (LS LOCATIONS 0A0 THROUGH 0BF)
:3859     CCR=0FC              ; CONSOLE READ REGISTER
:3860     TIMER=0FD           ; INTERVAL TIMER VALUE.
:3861     ALU.CC=0FE         ; ALU CONDITION CODES
:3862
:3863     THIS LOCATION IS A WRITE ONLY REGISTER. THE FOLLOWING BITS IN THE
:3864     PSL ARE AFFECTED:
:3865
:3866     COMPATIBILITY MODE - BIT<31>
:3867     CURRENT MODE - BITS<25:24>
:3868     INTERRUPT PRIORITY LEVEL (IPL) - BITS<20:16>
:3869     TRACE TRAP ENABLE (REALLY T OR TP) - BIT<4>
:3870     NEGATIVE CONDITION CODE - BIT<3>
:3871     ZERO CONDITION CODE - BIT<2>
:3872     OVERFLOW CONDITION CODE - BIT<1>
:3873     CARRY CONDITION CODE - BIT<0>
:3874
:3875     PSL.HW=OFF           ; HARDWARE PSL BITS
:3876
:3877
:3878 ; These addresses are only accessible with the MOV micro instruction
:3879 ; starting at address 80(H).
:3880
:3881     SETCSR.F0=80          ;SET IDC CSR FUNCTION BITS =0 & SET WRT INHIBIT
:3882     SETCSR.F1=81          ;SET IDC CSR FUNCTION BITS =1 & SET WRT INHIBIT
:3883     SETCSR.F2=82          ;SET IDC CSR FUNCTION BITS =2 & SET WRT INHIBIT
:3884     SETCSR.F3=83          ;SET IDC CSR FUNCTION BITS =3 & SET WRT INHIBIT
:3885     SETCSR.F4=84          ;SET IDC CSR FUNCTION BITS =4 & SET WRT INHIBIT
:3886     SETCSR.F5=85          ;SET IDC CSR FUNCTION BITS =5 & SET WRT INHIBIT
:3887     SETCSR.F6=86          ;SET IDC CSR FUNCTION BITS =6 & SET WRT INHIBIT
:3888     SETCSR.F7=87          ;SET IDC CSR FUNCTION BITS =7 & SET WRT INHIBIT
:3889     SETCSR.IE=88          ;SET INTERRUPT ENABLE IN THE CSR
:3890     SETCSR.CRDY=89        ;SET BIT <7> CRDY IN THE CSR
:3891     SETCSR.DS0=8A         ;SET CSR DRIVE SELECT BITS = 0
:3892     SETCSR.DS1=8B         ;SET CSR DRIVE SELECT BITS = 1
:3893     SETCSR.DS2=8C         ;SET CSR DRIVE SELECT BITS = 2
:3894     SETCSR.DS3=8D         ;SET CSR DRIVE SELECT BITS = 3
:3895     SETCSR.SSEI=8E        ;SET SKIP SECTOR ERROR INHIBIT IN THE CSR
:3896     SETCSR.SSFLG=8F       ;SET SKIP SECTOR FLAG IN THE CSR
:3897     SETCSR.MAINT=90        ;SET THE MAINTENANCE BIT IN THE CSR
:3898
:3899     CSR.MASK=91
:3900     C53FFC31.MASK=91      ;MASK TO CHECK WRITABLE BITS OF THE CSR
:3901
:3902     SAV.DATA.PAT=92        ;LOCATION TO STORE THE CURRENT DATA PATTERN
:3903
    
```

22-ENKCF-1.4 :
: ENKCF.MCR
: ENKCF.MIC

ENKCF.MIC

MICRO2 1M(01)

COMMON LS ASSIGNMENT 15-MAR-83

15-MAR-83 11:46:22

COMMON LS ASSIGNMENTS (TO REGISTERS)

8 8

Fiche 1 Frame B8

Sequence 92

Rev 01.40

Page 86

22.

:

:

```
:3904      CSR.DAT.PAT1=93      ;DATA PATTERN TO TEST THE IDC CSR WRITABLE BITS
:3905      CSR.DAT.PAT2=94      ;DATA PATTERN TO TEST THE IDC CSR WRITABLE BITS
:3906      CSR.DAT.PAT3=95      ;DATA PATTERN TO TEST THE IDC CSR WRITABLE BITS
:3907      CSR.DAT.PAT4=96      ;DATA PATTERN TO TEST THE IDC CSR WRITABLE BITS
:3908      CSR.DAT.PAT5=97      ;DATA PATTERN TO TEST THE IDC CSR WRITABLE BITS
:3909
:3910      #1A=98                ;CONSTANT
:3911
:3912
:3913      PORT0=99              ;RESERVED FOR PORT DATA
:3914      PORT1=9A              ;RESERVED FOR PORT DATA
:3915      PORT2=9B              ;RESERVED FOR PORT DATA
:3916      PORT3=9C              ;RESERVED FOR PORT DATA
:3917      PORT4=9D              ;RESERVED FOR PORT DATA
:3918      PORT5=9E              ;RESERVED FOR PORT DATA
:3919      PORT6=9F              ;RESERVED FOR PORT DATA
:3920      PORT7=0A0             ;RESERVED FOR PORT DATA
:3921      PORT8=0A1             ;RESERVED FOR PORT DATA
:3922      PORT9=0A2             ;RESERVED FOR PORT DATA
:3923      PORT10=0A3            ;RESERVED FOR PORT DATA
:3924      PORT11=0A4            ;RESERVED FOR PORT DATA
:3925      PORT12=0A5            ;RESERVED FOR PORT DATA
:3926      PORT13=0A6            ;RESERVED FOR PORT DATA
:3927      PORT14=0A7            ;RESERVED FOR PORT DATA
:3928      PORT15=0A8            ;RESERVED FOR PORT DATA
:3929      PORT16=0A9            ;RESERVED FOR PORT DATA
:3930      PORT17=0AA            ;RESERVED FOR PORT DATA
:3931      PORT18=0AB            ;RESERVED FOR PORT DATA
:3932      PORT19=0AC            ;RESERVED FOR PORT DATA
:3933      PORT20=0AD            ;RESERVED FOR PORT DATA
:3934      PORT21=0AE            ;RESERVED FOR PORT DATA
:3935      PORT22=0AF            ;RESERVED FOR PORT DATA
:3936      PORT23=0B0            ;RESERVED FOR PORT DATA
:3937      PORT24=0B1            ;RESERVED FOR PORT DATA
:3938      PORT25=0B2            ;RESERVED FOR PORT DATA
:3939      PORT26=0B3            ;RESERVED FOR PORT DATA
:3940      PORT27=0B4            ;RESERVED FOR PORT DATA
:3941      PORT28=0B5            ;RESERVED FOR PORT DATA
:3942      PORT29=0B6            ;RESERVED FOR PORT DATA
:3943      PORT30=0B7            ;RESERVED FOR PORT DATA
:3944      PORT31=0B8            ;RESERVED FOR PORT DATA
:3945      #0000FFFF=0B9         ;CONSTANT
:3946      FORCE.IDLE=0BA         ;DATA TO FORCE IDC FROM DIAG.IDLE TO IDLE
:3947      #FFF80000=0BB         ;MASK FOR DAR..LOOKS AT BITS <18:0>
:3948      DAR.MASK=0BB
:3949      CSR.PAT.1=0BC          ;DATA PAT 1 TO TEST CSR
:3950      CSR.PAT.2=0BD          ;DATA PAT 2 TO TEST CSR
:3951      CSR.PAT.3=0BE          ;DATA PAT 3 TO TEST CSR
:3952      CSR.PAT.4=0BF          ;DATA PAT 4 TO TEST CSR
:3953      CSR.PAT.5=0C0          ;DATA PAT 5 TO TEST CSR
:3954      BIT7.MASK=0C1          ;MASK TO CHECK ONLY BIT 7
:3955      BIT22.MASK=0C2         ;MASK TO CHECK ONLY BIT 22
:3956      BIT23.MASK=0C3         ;MASK TO CHECK ONLY BIT 23
:3957      #3=0C4                ;CONSTANT OF 3
:3958      #FE=0C5                ;CONSTANT OF FE
```

22-ENKCF-1.4 ;
: ENKCF.MCR
: ENKCF.MIC

ENKCF.MIC

MICRO2 1M(01)

COMMON LS ASSIGNMENTS (TO REGISTERS)

15-MAR-83 11:46:22

ENKCF Micro-diagnostic for IDC module

Sequence 93
Rev 01.40

Page 87

```

:3959      #1FE=0C6      ;CONSTANT OF 1FE
:3960      DRIVE.STATUS=0C7 ;LOCATION CONTAINS DRIVE INFORMATION
:3961      #FFFFFFF0=0C8 ;MASK FOR BITS<3:0>
:3962      #6=0C9      ;CONSTANT OF 6
:3963      #7FFFF=0CA ;CONSTANT
:3964      #F=0CB      ;CONSTANT
:3965      #FFFFFFCFF=0CC ;CONSTANT
:3966      #FFFFFFFC=0CD ;CONSTANT
:3967      #FFFFFFF80=0CE ;CONSTANT
:3968      SETCSR.R80=0CF ;SETS BIT 29
:3969      #20100500=0D0 ;CONSTANT
:3970      #A0100500=0D1 ;CONSTANT
:3971      #FFCFFFFFFF=0D2 ;MASK TO CHECK BITS<21:20>
:3972      #FFFFE000=0D3 ;MASK TO CHECK BITS<12:0>
:3973      #69=0D4      ;CONSTANT = 105
:3974      #6837=0D5 ;CONSTANT
:3975      #7200=0D6 ;CONSTANT
:3976      #FFFFFF800=0D7 ;MASK TO CHECK BITS <10:0>
:3977      #1021=0D8 ;CONSTANT
:3978      #A0010500=0D9 ;CONSTANT
:3979      #20010500=0DA ;CONSTANT
:3980      #1F=0DB ;CONSTANT
:3981      #169=0DC ;CONSTANT
:3982      #101=0DD ;CONSTANT
:3983      #FFF0FFFF=0DE ;MASK TO CHECK BITS <19:16>
:3984      #30000=0DF ;CONSTANT
:3985      #70000=0E0 ;CONSTANT
:3986      #F0000=0E1 ;CONSTANT
:3987      #FFFFEBFF=0E2 ;MASK TO CHECK BITS <12> <10>
:3988      #FFFF6BFF=0E3 ;MASK TO CHECK BITS <15> <12> <10>
:3989      #9400=0E4 ;CONSTANT
:3990      #FFFF6FFF=0E5 ;MASK TO CHECK BITS <15> <12>
:3991      #9000=0E6 ;CONSTANT
:3992      #FFFFFFF=0E7 ;MASK TO CHECK BIT <12>
:3993      #11111111=0E8 ;CONSTANT
:3994      #22222222=0E9 ;CONSTANT
:3995      #44444444=0EA ;CONSTANT
:3996      #000F0000=0EB ;CONSTANT
:3997      #2D35E=0EC ;CONSTANT
:3998      #28AEE=0ED ;CONSTANT
:3999      #5A6BF=0EE ;CONSTANT
:4000      #5A6C2=0EF ;CONSTANT
:4001      #FFFFFBFF=0F0 ;MASK TO CHECK BIT <10>
:4002      #FFFFFF7F=0F1 ;MASK TO CHECK BIT <7>
:4003      #10000080=0F2 ;BITS TO DISABLE TIMEOUTS AND SET CSR<CRDY>
:4004      #B4D84=0F3 ;WAIT LOOP TO WAIT 400ms
:4005      #FF3FFFFFF=0F4 ;MASK TO CHECK <23:22>
:4006      #FFFFFFE3=0F5 ;MASK TO CHECK <4:2>
:4007      #FEFFFFFF=0F6 ;MASK TO CHECK <24>
:4008
:4009
```

```

:4010 .page 'Microinstruction Formats'
:4011
:4012     The following is taken directly from the NEBULA Hardware
:4013     Specification.
:4014
:4015             MICROINSTRUCTION FORMATS
:4016
:4017
:4018
:4019
:4020             22 21      16 15      9 8 7 6 5 4      0
:4021             +-----+-----+-----+-----+
:4022 1. BASIC      |1|      DP      |      D ADRS      | B | CC | SCTL |
:4023             +-----+-----+-----+-----+
:4024
:4025             22 20 19 17 16      9 8 7 6 5 4      0
:4026             +-----+-----+-----+-----+
:4027 2. MOVE      |0 1 1| MDP |      XD ADRS      | B | CC | SCTL |
:4028             +-----+-----+-----+-----+
:4029
:4030             22 20 19      14 13 11  9 8 7 6 5 4      0
:4031             +-----+-----+-----+-----+
:4032 3. EXTENDED  |0 1 0|      XDP      | SI | A | B | CC | SCTL |
:4033             +-----+-----+-----+-----+
:4034
:4035             22      19 18 16 15      9 8 7 6 5 4      0
:4036             +-----+-----+-----+-----+
:4037 4. MEM.REQ    |0 0 1 1| MF1 |      D ADRS      | MF2| DT | SCTL |
:4038             +-----+-----+-----+-----+
:4039
:4040             22      19 18 16 15  12 11  9 8 7 6 5 4      0
:4041             +-----+-----+-----+-----+
:4042 5. MISC      |0 0 1 0| P|0 0|      M1 | M2 |0|R|0 0| SCTL |
:4043             +-----+-----+-----+-----+
:4044
:4045             22      19 18 17      4 3      0
:4046             +-----+-----+-----+-----+
:4047 6. JUMP      |0 0 0 1| QS|      JUMP ADDRESS      | JCTL |
:4048             +-----+-----+-----+-----+
:4049
:4050             22      19 18 17      12 11  9 8 7 6 5 4 3      0
:4051             +-----+-----+-----+-----+
:4052 7. DECODE     |0 0 0 0|      DEC.ADRS      | IFUNC| | | | JCTL |
:4053             +-----+-----+-----+-----+
:4054             |
:4055             BACK UP PC      IB REQ---+ | | | | ---OPC/SPEC
:4056             |
:4057             SEL CM HI IR BYTE---+ | | | | ---LOAD RDEST
:4058             |
:4059             |
             LOAD OS---+

```

```

:4060 .page
:4061 :
:4062 :
:4063 1. 'BASIC' Microinstruction
:4064 :
:4065 CSR<22> = 1 (OPCODE)
:4066 :
:4067 This opcode bit causes LS Adrs <7> to be cleared.
:4068 :
:4069 CSR<21:16> (Data Path Control)
:4070 :
:4071 This field controls the data path. It controls the 2901 ALU,
:4072 the ALU data source, the data destination in the 2901 array,
:4073 and the write to the Local Store.
:4074 :
:4075 CSR<15:9> (D Adrs <6:0>)
:4076 :
:4077 This field selects which of the 128 accessible Local Store
:4078 locations to use.
:4079 :
:4080 CSR<8:7> (B Adrs)
:4081 :
:4082 This field selects one of the four Working Registers.
:4083 :
:4084 CSR<6:5> (Condition Codes)
:4085 :
:4086 This field specifies the data type and condition code control.
:4087 :
:4088 CSR<4:0> (SCTL)
:4089 :
:4090 This field specifies the skip condition/micro sequencer
:4091 function.

```



```

:4092 .page
:4093
:4094
:4095 2. 'MOVE' Microinstruction
:4096
:4097     CSR<22:20> = 011 (OPCODE)
:4098
:4099     This combination of opcode bits allows CSR<16> to control
:4100     LS Adrs <7>. This permits access of LS locations 80 - FF.
:4101
:4102     CSR<19:17> (Data Path Control)
:4103
:4104     This field is decoded to produce some data path control
:4105     information.
:4106
:4107         0 LS[XD] <- WR[B], WR[B] <- MEM DATA* 4 LS[XD] <- MEM DATA*
:4108         1 MEMORY <- LS [XD]*                    5 LS[XD] <- ACC/PORT
:4109         2 Q <- XWR[B]                            6 LS[XD] <- WR[B] + OS
:4110         3 WR[B] <- LS [XD]                        7 LS[XD] <- WR[B]
:4111
:4112     CSR<16:9> (XD<7:0>)
:4113
:4114     This field specifies which of the 256 Local Store locations to
:4115     use.
:4116
:4117     CSR<8:7> (B Adrs)
:4118
:4119     This field specifies which of the four Working Registers to
:4120     use. For MDP = 2, this field selects either the Backup PC or
:4121     the VAR.
:4122
:4123     CSR<6:5> (CC)
:4124
:4125     This field specifies the data type and condition code control.
:4126
:4127     CSR<4:0> (SCTL)
:4128
:4129     This field specifies the Skip control. See Table 3.
:4130
:4131
:4132 * For information on which Working Registers and Local Store locations
:4133   may be used for Memory Addresses and data source/destinations, see
:4134   the Memory Management documentation. Note that these restrictions
:4135   are due to microcoding conventions and not hardware.
    
```

```

:4136 .page
:4137 :
:4138 :
:4139 3. 'EXTENDED' Microinstruction
:4140 :
:4141 CSR<22:20> = 010 (OPCODE)
:4142 :
:4143 This opcode causes a function internal to the 2901 array; that
:4144 is, an operation involving only Working Registers and the Q-
:4145 Register.
:4146 :
:4147 CSR<19:14> (Data Path Control)
:4148 :
:4149 This field is decoded to supply the 2901 ALU function code,
:4150 the ALU Source control, the destination code, and the ALU
:4151 Carry-In.
:4152 :
:4153 CSR<13:11> (Shift Input)
:4154 :
:4155 This field selects the data to be shifted into a register,
:4156 when a Shift function is being done.
:4157 :
:4158 CSR<10:9> (A Adrs)
:4159 :
:4160 This field selects which Working Register to use as a source
:4161 of data, when RAM A is selected to the ALU.
:4162 :
:4163 CSR<8:7> (B Adrs)
:4164 :
:4165 This field selects which Working Register to use as a source
:4166 (when RAM B is selected to the ALU) and/or destination (when
:4167 the RAM is written) of data.
:4168 :
:4169 CSR<6:5> (CC)
:4170 :
:4171 This field specifies the data type and condition code control.
:4172 :
:4173 CSR<4:0> (SCTL)
:4174 :
:4175 This field specifies the skip condition/micro sequencer
:4176 function.
    
```

```

:4177 .page
:4178
:4179
:4180 4. 'MEM.REQ' Microinstruction
:4181
:4182     CSR<22:19> = 0011     (OPCODE)
:4183
:4184         This opcode causes a Memory Request to be started. The Local
:4185         Store is output on on the D-Bus, to be used as the virtual
:4186         address. Working Register 5 is written with this address.
:4187
:4188     CSR<18:16>, <8:7>     (Memory Function)
:4189
:4190         This field specifies to the Memory Controller the type of
:4191         request: physical, virtual, type of access, etc. For the
:4192         list of functions and the codes, see the NEBULA Memory Spec.
:4193
:4194     CSR<15:9>     (D Adrs <6:0>)
:4195
:4196         This field selects which of the 128 accessible Local Store
:4197         locations to use as the virtual address.
:4198
:4199     CSR<6:5>     (Data Type)
:4200
:4201         This field specifies the data type of the request.
:4202
:4203             00:=BYTE
:4204             01:=WORD
:4205             10:=SIZE Reg
:4206             11:=LONG
:4207
:4208     CSR<4:0>     (SCTL)
:4209
:4210         This field specifies the skip condition/micro sequencer
:4211         function.
    
```



```

:4265 .page
:4266 :
:4267 : CSR<8> (MUST BE ZERO)
:4268 :
:4269 : CSR<7> (WR Address)
:4270 :
:4271 :
:4272 : This bit selects WR[0] or WR[1] to be put on the Y-Bus.
:4273 :
:4274 : CSR<6:5> (Not Used)
:4275 :
:4276 : CSR<4:0> (SCTL)
:4277 :
:4278 : This field specifies the skip condition/micro sequencer
:4279 : function.
:4280 :
    
```

U C

U C

U C

U C

U C

U C

U C

U C

U C

U C

U C

```

:4281 .page
:4282
:4283
:4284 6. "JUMP" Microinstruction
:4285
:4286 CSR<22:19> = 0001 (OPCODE)
:4287
:4288 This opcode causes the data path to no-op, and the micro
:4289 sequencer to execute a JUMP.
:4290
:4291 CSR<18> (Select OS)
:4292
:4293 This bit, when asserted, causes OS<4:0> to be OR'ed with the
:4294 five low bits of the of the jump microaddress.
:4295
:4296 CSR<17:4> (Jump Microaddress)
:4297
:4298 If Select OS is CLEAR, this field specifies the fourteen-bit
:4299 jump micro address. If Select OS is SET, CSR<17:9> is Jump
:4300 Adrs<13:5> and OS<4:0> OR'ed with CSR<8:4> is Jump Adrs<4:0>.
:4301
:4302 CSR<3:0> (JCTL)
:4303
:4304 This field specifies the jump condition/micro sequencer
:4305 function.

```

```

4306 .page
4307 .
4308 .
4309 7. 'DECODE' Microinstruction
4310 .
4311 A. DESCRIPTION
4312 .
4313 CSR<22:19> = 0000 (OPCODE)
4314 .
4315 This combination of opcode bits causes the Local Store to
4316 access location 10 (the PC.) The Local Store drives the D-Bus.
4317 The 2901 is set up to input the data on the D-Bus, increment
4318 it, and output to the Y-Bus. Thus, the Y-Bus contains the PC
4319 + 1.
4320 .
4321 CSR<18> (BPC [asserted low])
4322 .
4323 If this bit is CLEAR, the incremented PC is written into the
4324 2901 RAM. If it is SET, the RAM is not written.
4325 .
4326 CSR<17:12> (DEC.ADRS <13:8>)
4327 .
4328 This data is taken to be the upper six bits of the next
4329 microaddress. The low eight bits are supplied by the IRD ROM.
4330 The actual JUMP/JSR/NO-OP is determined by the JCTL field.
4331 .
4332 CSR<11:9> (IFUNC)
4333 .
4334 This is the IFUNC field. It defines which fork is being
4335 taken.
4336 .
4337 CSR<8> (IB.REQ)
4338 .
4339 This is the IB.REQ bit. When it is CLEAR, the LS is not
4340 written, no interaction with memory is initiated and the
4341 instructin buffer is not affected.
4342 .
4343 When it is SET:
4344 .
4345 The incremented PC (valid on the Y-Bus by the end of the
4346 cycle) is re-written to Local Store location 0. This
4347 completes the PC increment function.
4348 .
4349 A Conditional Memory Request is executed. If the two low bits
4350 of the PC are not 11, the request is not initiated. If the
4351 two low bits are set, an IB PREFETCH is done. The
4352 (unincremented) PC is output to the memory and the CPU grant
4353 (CPUG) signal is examined. If CPUG is low by T120, the CPU
4354 will stall until the CPUG signal becomes true. After the
4355 request is completed, the next state entered is dependent upon
4356 the JCTL field.

```

[illegible]

```

:4357 page
:4358
:4359 (CSR<7> (SEL CM HI IR BYTE)
:4360
:4361 This bit enables the upper byte of the Compatibility Mode
:4362 instruction in the Prefetch register to drive the IB-Bus. It
:4363 is used only while in Compatibility Mode, when IRD is being
:4364 done. It must never be asserted in VAX Mode.
:4365
:4366
:4367 (CSR<6> (LD.OS)
:4368
:4369 If this bit is SET, the eight-bit data on the IB Bus (the
:4370 output of the instruction buffer) is loaded into the OS
:4371 register. This load is dependent upon Compatibility Mode and
:4372 LD R Dest (If CM.AND.LD R Dest, OS<3> <- 0.) If the bit is
:4373 CLEAR, OS is not affected.
:4374
:4375 (CSR<5> (LD R DEST)
:4376
:4377 If this bit is SET, the R Dest Skip bit will be SET for
:4378 Specifier Mode equal to 5 and CLEARED for Mode not equal to 5,
:4379 in VAX mode, or mode 0 while in Compatibility Mode. If this
:4380 bit is CLEAR, the R Dest bit will be unaffected.
:4381
:4382 (CSR<4> (OPC/SPEC)
:4383
:4384 This bit (effectively another IFUNC bit) defines the data to
:4385 be decoded; that is, if it is an Opcode or a Specifier. In
:4386 hardware, it selects which ROM is to drive the micro address
:4387 bus.
:4388
:4389 (CSR<3:0> (JCTL)
:4390
:4391 These four bits are decoded to make one of 16 functions. See
:4392 Table 3.

```


U OC
U OC

U OC
U OC

U OC
U OC

U OC
U OC

U OC
U OC

U OC
U OC

U OC
U OC

U OC
U OC

U	OC
U	OC

U OC
U OC

U	OC
U	OC

U	OC
U	OC

U	OC
U	OC

U OC
U OC

U OC
U OC

U OC
U OC

:4439
:4440
:4441
:4442
:4443
:4444
:4445
:4446
:4447
:4448
:4449
:4450
:4451
:4452
:4453
:4454
:4455
:4456
:4457
:4458
:4459
:4460
:4461
:4462
:4463
:4464
:4465
:4466
:4467
:4468
:4469
:4470
:4471
:4472
:4473
:4474
:4475
:4476
:4477
:4478

page

Table 2. Local Store Write

The Local Store Write Controller examines CSR<22:17>, CSR<6>, the SIZE register (SIZE<1:0>) and the Compatibility Mode bit (CM). These eight bits are decoded to determine which parts of the LS to write, if any. The LS is written with data from the 2901 ALU (Y-Bus).

CSR	22	21	20	19	18	17	
1. BASIC	1	0	X	X	X	X	No Write
2. MOVE	1	1	X	X	X	X	Write *
	0	1	1	0	X	1	No Write
	0	1	1	0	1	X	No Write
	0	1	1	1	X	X	Write *
3. EXTENDED	0	1	1	X	X	X	No Write
4. MEM.REQ	0	0	1	1	X	X	No Write
5. MISC	0	0	1	0	X	X	No Write
6. JUMP	0	0	0	1	X	X	No Write
7. DECODE	0	0	0	0	X	X	Conditional Write **

* LS Write, Data Type Dependent. These cases now examine CSR<6>.

If CSR<6> = 0, and CM = 0 Write LS<31:0> (Data Type = LONG)

If CSR<6> = 0, and CM = 1 Write LS<15:0> (Compatibility Mode. DT = WORD)

If CSR<6> = 1, the Write is conditional on the SIZE register.

If CM = 0 SIZE<1:0> = 00 Write LS<7:0> (BYTE)

= 01 Write LS<15:0> (WORD)

= 1X Write LS<31:0> (LONG)

If CM = 1 Write LS<15:0> (WORD)

** If CSR<8> = 0, No Write

If CSR<8> = 1, Write LS<31:0> (Writes incremented PC)

:4479 .page

:4480

:4481

:4482

:4483

:4484

:4485

:4486

:4487

:4488

:4489

:4490

:4491

:4492

:4493

:4494

:4495

:4496

:4497

:4498

:4499

:4500

:4501

:4502

:4503

:4504

:4505

:4506

:4507

:4508

:4509

:4510

:4511

:4512

:4513

:4514

:4515

:4516

:4517

:4518

:4519

:4520

:4521

:4522

:4523

.page

Table 3. SKIP/JUMP Conditions

This table summarizes the Skip and Jump conditions, and the other special functions of the micro sequencer.

SCTL codes 0-F, 17-1F are Skip conditions. The codes 10-17 execute special functions. JCTL codes 0-8 are Jump conditions, and C-F execute special functions.

Sctl	Function	Sctl	Function
00	Not ALU N	10	RTN+1 if No Mem Err
01	Not ALU Z	11	Loop if Not ALU Z
02	Not ALU V	12	POP Stack
03	ALU C	13	Loop if ALU C
04	Not PSL C	14	Return
05	Branch False	15	No Skip (NOP)
06	RDEST	16	RTN+1
07	Not Inter. Req.	17	Loop if No Inter. and No Sync
08	ALU N	18	Console ATTN
09	ALU Z	19	Console ACK
0A	ALU V	1A	Port Interrupt Pending
0B	Not ALU C	1B	Reg. Backup Mask Flag
0C	State 0	1C	ALU N.XOR.ALU V
0D	State 1	1D	Not Memory Error
0E	Not State 0	1E	Skip
0F	Not State 1	1F	Sync

Jctl	Function	Jctl	Function
00	Not ALU N	08	ALU N
01	Not ALU Z	09	ALU Z
02	Not ALU V	0A	ALU V
03	ALU C	0B	Not ALU C
04	JUMP	0C	JSR
05	Branch False	0D	JSR if IB Valid
06	RDEST	0E	No Jump; ip if IB Valid
07	Not Interr. Req	0F	IB Valid

:4524 .PAGE '2901 ALU Control Description'
:4525
:4526
:4527
:4528
:4529
:4530
:4531
:4532
:4533
:4534
:4535
:4536
:4537
:4538
:4539
:4540
:4541
:4542
:4543
:4544
:4545
:4546
:4547
:4548
:4549
:4550
:4551
:4552
:4553
:4554
:4555
:4556
:4557
:4558
:4559
:4560
:4561
:4562
:4563
:4564
:4565
:4566
:4567
:4568
:4569
:4570
:4571

The 2901 ALU is the heart of the CPU module. It is responsible for all CPU calculations. The control lines going to I0 through I8 control the function, input, and output of the 2901. When a failure involving the 2901 occurs, the logical place to start looking is the control lines. The following tables describe what the control signals should be for various functions. Use it whenever the control lines are to be checked. The test descriptions tell the operator what function the 2901 is supposed to be doing.

The SRC CTL on lines I0 through I2 (pins 12 through 14 respectively) control where the inputs to the internal ALU within the 2901 come from. The inputs to the internal ALU are labeled R and S. R is one operand (4 bits) and S is the other operand (also 4 bits). The R input can come from a Working Register (labeled A), the D bus (direct input), or be forced to 0. The S input can come from either set of working registers (labeled A and B), the Q register, or forced to 0. A table of these control signals follow:

			octal code	SRC operands	
I2	I1	I0		R	S
L	L	L	0	A	Q
L	L	H	1	A	B
L	H	L	2	0	Q
L	H	H	3	0	B
H	L	L	4	0	A
H	L	H	5	D	A
H	H	L	6	D	Q
H	H	H	7	D	0

ALU SOURCE OPERAND CONTROL

The ALU CTL on lines I3 through I5 (pins 26, 28, 27 respectively) control what function the ALU is to perform on the two operands R and S. A table follows:

4572
4573
4574
4575
4576
4577
4578
4579
4580
4581
4582
4583
4584
4585
4586
4587
4588
4589
4590
4591
4592
4593
4594
4595
4596
4597
4598
4599
4600
4601
4602
4603
4604
4605
4606
4607
4608
4609
4610
4611
4612
4613
4614
4615
4616
4617
4618
4619
4620
4621
4622
4623
4624
4625

page

15	14	13	octal code	ALU function
L	L	L	0	R PLUS S
L	L	H	1	S MINUS R
L	H	L	2	R MINUS S
L	H	H	3	R OR S
H	L	L	4	R AND S
H	L	H	5	R NOT AND S
H	H	L	6	R XOR S
H	H	H	7	R XNOR S

ALU FUNCTION CONTROL

The DST CTL on lines 16 through 18 (pins 5, 7, 6 respectively) control what goes out onto the Y bus (either the internal ALU result or a WR directly) and what happens internal to the 2901 (shifting, writing the Q register etc). Internal shifting left or right is accomplished with this control. Below is a table explaining the various destination possibilities. The arrow refers to where the output will end up. B indicates the WR addressed by the B ADDR, F refers to the output of the internal ALU, and A refers to the output of the WR addressed by the A ADDR. Up is toward the MSB, while down is toward the LSB.

nomenclature in listing	18	17	16	octal code	RAM function shift	load	Q-reg shift	function load	Y output
LOAD Q	L	L	L	0	NONE	NONE	NONE	F->Q	F
NOP	L	L	H	1	NONE	NONE	NONE	NONE	F
WRITE.B.A	L	H	L	2	NONE	F->B	NONE	NONE	A
WRITE.B.F	L	H	H	3	NONE	F->B	NONE	NONE	F
RSHF.RAM.Q	H	L	L	4	DOWN	F/2->B	DOWN	Q/2->Q	F
RSHF.RAM	H	L	H	5	DOWN	F/2->B	NONE	NONE	F
LSHF.RAM.Q	H	H	L	6	UP	2F->B	UP	2Q->Q	F
LSHF.RAM	H	H	H	7	UP	2F->B	NONE	NONE	F

ALU DESTINATION CONTROL

:4656 .PAGE "TEST POINTERS"

:4657
 :4658 This portion contains the pointers to all tests in this section. The
 :4659 WCS address of the JUMP instruction corresponds to the test number.
 :4660 E.g. WCS 5 contains a JUMP to test 5. All unused test numbers will
 :4661 contain a JUMP to an error routine. This portion is 80. locations
 :4662 long, allowing for up to 80. tests per section, from WCS address 1 to
 :4663 WCS address 4F.
 :4664
 :4665
 :4666
 :4667
 :4668

:TEST POINTERS BEGIN AT WCS ADDRESS 1

U 0001, 8875,74	:4669	JMP [T.1]	:GO TO TEST 1
U 0002, 8878,64	:4670	JMP [T.2]	:GO TO TEST 2
U 0003, 087B,E4	:4671	JMP [T.3]	:GO TO TEST 3
U 0004, 087E,44	:4672	JMP [T.4]	:GO TO TEST 4
U 0005, 0884,84	:4673	JMP [T.5]	:GO TO TEST 5
U 0006, 088A,A4	:4674	JMP [T.6]	:GO TO TEST 6
U 0007, 088F,A4	:4675	JMP [T.7]	:GO TO TEST 7
U 0008, 8893,A4	:4676	JMP [T.8]	:GO TO TEST 8
U 0009, 8896,C4	:4677	JMP [T.9]	:GO TO TEST 9
U 000A, 0899,D4	:4678	JMP [T.A]	:GO TO TEST A
U 000B, 089E,54	:4679	JMP [T.B]	:GO TO TEST B
U 000C, 08A0,24	:4680	JMP [T.C]	:GO TO TEST C
U 000D, 88A3,64	:4681	JMP [T.D]	:GO TO TEST D
U 000E, 88A7,24	:4682	JMP [T.E]	:GO TO TEST E
U 000F, 88AE,D4	:4683	JMP [T.F]	:GO TO TEST F
U 0010, 08B4,B4	:4684	JMP [T.10]	:GO TO TEST 10
U 0011, 08BB,74	:4685	JMP [T.11]	:GO TO TEST 11
U 0012, 08C1,F4	:4686	JMP [T.12]	:GO TO TEST 12
U 0013, 88C9,54	:4687	JMP [T.13]	:GO TO TEST 13
U 0014, 08D0,A4	:4688	JMP [T.14]	:GO TO TEST 14
U 0015, 08D8,D4	:4689	JMP [T.15]	:GO TO TEST 15
U 0016, 88E0,B4	:4690	JMP [T.16]	:GO TO TEST 16
U 0017, 88E9,84	:4691	JMP [T.17]	:GO TO TEST 17
U 0018, 88F1,D4	:4692	JMP [T.18]	:GO TO TEST 18
U 0019, 08F7,64	:4693	JMP [T.19]	:GO TO TEST 19
U 001A, 88FB,B4	:4694	JMP [T.1A]	:GO TO TEST 1A
U 001B, 08FF,44	:4695	JMP [T.1B]	:GO TO TEST 1B
U 001C, 090D,44	:4696	JMP [T.1C]	:GO TO TEST 1C
U 001D, 8913,34	:4697	JMP [T.1D]	:GO TO TEST 1D
U 001E, 0917,C4	:4698	JMP [T.1E]	:GO TO TEST 1E
U 001F, 091E,94	:4699	JMP [T.1F]	:GO TO TEST 1F
U 0020, 8925,54	:4700	JMP [T.20]	:GO TO TEST 20
U 0021, 0928,94	:4701	JMP [T.21]	:GO TO TEST 21
U 0022, 092A,74	:4702	JMP [T.22]	:GO TO TEST 22
U 0023, 892F,C4	:4703	JMP [T.23]	:GO TO TEST 23
U 0024, 8935,14	:4704	JMP [T.24]	:GO TO TEST 24
U 0025, 893A,24	:4705	JMP [T.25]	:GO TO TEST 25
U 0026, 893C,D4	:4706	JMP [T.26]	:GO TO TEST 26
U 0027, 0940,B4	:4707	JMP [T.27]	:GO TO TEST 27
U 0028, 8943,F4	:4708	JMP [T.28]	:GO TO TEST 28
U 0029, 8947,D4	:4709	JMP [T.29]	:GO TO TEST 29
U 002A, 094B,94	:4710	JMP [T.2A]	:GO TO TEST 2A

ZZ-ENKCF-1.4	:	ENKCF.MIC	TEST POINTERS	15-MAR-83	Fiche 1 Frame 19	Sequence 112	
:	:	MICRO2 1M(01)	15-MAR-83 11:46:22	ENKCF Micro-diagnostic for IDC module	Rev 01.40	Page 106	
:	:	TEST POINTERS					

U 002B, 8950,44	:4711	JMP [T.2B]	:GO TO TEST 2B
U 002C, 0957,D4	:4712	JMP [T.2C]	:GO TO TEST 2C
U 002D, 095E,44	:4713	JMP [T.2D]	:GO TO TEST 2D
U 002E, 0963,A4	:4714	JMP [T.2E]	:GO TO TEST 2E
U 002F, 8968,54	:4715	JMP [T.2F]	:GO TO TEST 2F
U 0030, 096C,54	:4716	JMP [T.30]	:GO TO TEST 30
U 0031, 0973,84	:4717	JMP [T.31]	:GO TO TEST 31
U 0032, 8978,D4	:4718	JMP [T.32]	:GO TO TEST 32
U 0033, 897F,04	:4719	JMP [T.33]	:GO TO TEST 33
U 0034, 8986,34	:4720	JMP [T.34]	:GO TO TEST 34
U 0035, 098C,84	:4721	JMP [T.35]	:GO TO TEST 35
U 0036, 8991,F4	:4722	JMP [T.36]	:GO TO TEST 36
U 0037, 8995,B4	:4723	JMP [T.37]	:GO TO TEST 37
U 0038, 099A,A4	:4724	JMP [T.38]	:GO TO TEST 38
U 0039, 899D,F4	:4725	JMP [T.39]	:GO TO TEST 39
U 003A, 09AA,A4	:4726	JMP [T.3A]	:GO TO TEST 3A
U 003B, 8981,E4	:4727	JMP [T.3B]	:GO TO TEST 3B
U 003C, 09B5,D4	:4728	JMP [T.3C]	:GO TO TEST 3C
U 003D, 89B9,F4	:4729	JMP [T.3D]	:GO TO TEST 3D
U 003E, 886B,E4	:4730	JMP [ERR.NO.TEST]	:GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:4731		: IN THIS OVERLAY
U 003F, 886B,E4	:4732	JMP [ERR.NO.TEST]	:GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:4733		: IN THIS OVERLAY
U 0040, 886B,E4	:4734	JMP [ERR.NO.TEST]	:GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:4735		: IN THIS OVERLAY
U 0041, 886B,E4	:4736	JMP [ERR.NO.TEST]	:GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:4737		: IN THIS OVERLAY
U 0042, 886B,E4	:4738	JMP [ERR.NO.TEST]	:GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:4739		: IN THIS OVERLAY
U 0043, 886B,E4	:4740	JMP [ERR.NO.TEST]	:GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:4741		: IN THIS OVERLAY
U 0044, 886B,E4	:4742	JMP [ERR.NO.TEST]	:GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:4743		: IN THIS OVERLAY
U 0045, 886B,E4	:4744	JMP [ERR.NO.TEST]	:GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:4745		: IN THIS OVERLAY
U 0046, 886B,E4	:4746	JMP [ERR.NO.TEST]	:GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:4747		: IN THIS OVERLAY
U 0047, 886B,E4	:4748	JMP [ERR.NO.TEST]	:GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:4749		: IN THIS OVERLAY
U 0048, 886B,E4	:4750	JMP [ERR.NO.TEST]	:GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:4751		: IN THIS OVERLAY
U 0049, 886B,E4	:4752	JMP [ERR.NO.TEST]	:GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:4753		: IN THIS OVERLAY
U 004A, 886B,E4	:4754	JMP [ERR.NO.TEST]	:GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:4755		: IN THIS OVERLAY
U 004B, 886B,E4	:4756	JMP [ERR.NO.TEST]	:GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:4757		: IN THIS OVERLAY
U 004C, 886B,E4	:4758	JMP [ERR.NO.TEST]	:GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:4759		: IN THIS OVERLAY
U 004D, 886B,E4	:4760	JMP [ERR.NO.TEST]	:GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:4761		: IN THIS OVERLAY
U 004E, 886B,E4	:4762	JMP [ERR.NO.TEST]	:GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:4763		: IN THIS OVERLAY
U 004F, 886B,E4	:4764	JMP [ERR.NO.TEST]	:GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:4765		: IN THIS OVERLAY

ZZ-ENKCF-1.4 :
: ENKCF.MCR
: ENKCF.MIC

ENKCF.MIC

MICRO2 1M(01)
DIAGNOSTIC POINTERS

DIAGNOSTIC POINTERS

15-MAR-83 11:46:22

Fiche 1 Frame J9

ENKCF Micro-diagnostic for IDC module

Sequence 113
Rev 01.40

Page 107

```
:4766 PAGE 'DIAGNOSTIC POINTERS'
:4767
:4768 This section contains all the pointers needed for this diagnostic
:4769 overlay. The first pointer (at WCS location 0) points to a data
:4770 transfer routine. It is the first location executed after a new
:4771 WCS load. The instruction here is a JUMP to TRANSFER.POINT which may
:4772 contain instructions to transfer data. If no data is to be trans-
:4773 ferred, or at the completion of the data transfers, the CPU will
:4774 issue CPU ATTN with all bits of the control and status word clear.
:4775
:4776 The next 80. locations (from WCS location 1 to WCS location 4F)
:4777 contain pointers to each test in this overlay. The pointers are
:4778 JUMP instructions to the test number corresponding to the location
:4779 number. E.g. location 5 will contain a JUMP to test 5. All unused
:4780 test numbers will contain a JUMP to an error routine. This portion
:4781 appears after the definitions in this listing.
:4782
:4783 Finally the next 32. locations (from WCS location 50 to 6F) contain
:4784 pointers (Jump instructions) to WCS subroutines which are to be used by
:4785 the console diagnostic monitor.
:4786
:4787
U 0000, 086C,24 :4788 0: JMP [TRANSFER.POINT] ;GO TO ROUTINE THAT LOADS LS 7 WITH
:4789 ; POINTER TO DATA SECTION AND ISSUES
:4790 ; CPU ATTN WITH TRANSFER BIT SET.
:4791
:4792 50: ;START AT WCS LOCATION 50 FOR SUB-
:4793 ; ROUTINE POINTERS
:4794
:4795 ; SUBROUTINE POINTERS
:4796
U 0050, 0860,84 :4797 JMP [DEPOSIT.CSR] ;GO TO WCS SUBROUTINE WHICH WRITES THE
:4798 ; MCT CSR FOR DEPOSIT CSR COMMAND
U 0051, 0860,D4 :4799 JMP [EXAMINE.CSR] ;GO TO WCS SUBROUTINE WHICH READS THE
:4800 ; MCT CSR FOR EXAMINE CSR COMMAND
U 0052, 0861,F4 :4801 JMP [DEPOSIT.TB] ;GO TO WCS SUBROUTINE WHICH WRITES THE
:4802 ; MCT TRANSLATION BUFFER FOR DEPOSIT
:4803 ; TB COMMAND
U 0053, 8863,A4 :4804 JMP [EXAMINE.TB] ;GO TO WCS SUBROUTINE WHICH READS THE
:4805 ; MCT TRANSLATION BUFFER FOR EXAMINE
:4806 ; TB COMMAND
U 0054, 8865,C4 :4807 JMP [DEPOSIT.MM] ;GO TO WCS SUBROUTINE WHICH WRITES MAIN
:4808 ; MEMORY FOR DEPOSIT MM COMMAND. ALSO
:4809 ; USED TRANSFERING DATA FROM WCS TO
:4810 ; MAIN MEMORY.
U 0055, 0866,24 :4811 JMP [EXAMINE.MM] ;GO TO WCS SUBROUTINE WHICH READS MAIN
:4812 ; MEMORY FOR EXAMINE MM COMMAND.
U 0056, 0868,94 :4813 JMP [SAVE.WR] ;GO TO WCS SUBROUTINE WHICH STORES THE
:4814 ; CONTENTS OF WRO-WR3 IN LS 0-3
U 0057, 8868,E4 :4815 JMP [RESTORE.WR] ;GO TO WCS SUBROUTINE WHICH RESTORES THE
:4816 ; CONTENTS OF WRO-WR3 FROM LS 0-3
U 0058, 8864,24 :4817 JMP [DEPOSIT.UBS] ;GO TO WCS SUBROUTINE WHICH WRITES THE
:4818 ; UNIBUS MAP ON THE MEMORY CONTROLLER
U 0059, 0865,44 :4819 JMP [EXAMINE.UBS] ;GO TO WCS SUBROUTINE WHICH READS THE
:4820 ; UNIBUS MAP ON THE MEMORY CONTROLLER
```


:4854 .PAGE
 :4855 .REGION/70,5FF
 :4856 .TOC 'LS DATA SECTION'
 :4857

:4858 :+++

:4859 This section lists the data that will be transfered to LS by the con-
 :4860 sole as part of the initialization performed in test 0. The TRANSFER
 :4861 DATA routine will point to this data area. LS contains the constants,
 :4862 control and status word, and temporary storage locations used by the
 :4863 tests and subroutines of the microdiagnostic.

:4864 The LS is 32 bits wide. Each of these words is 16 bits, so two con-
 :4865 secutive words listed here will be loaded into each consecutive LS loc-
 :4866 ation. The first word listed will be bits 0-15 of the LS location, the
 :4867 second word listed will be bits 16-31 of the same LS location.

:4868 The locations 78-7F in the first half of LS and F8-FF in the second
 :4869 half of LS are not actual LS locations. They force an access to one of
 :4870 several registers in the CPU or force an indexing feature to another LS
 :4871 location. For that reason, they are not written via the transfer rou-
 :4872 tine.

:4873 Note: This table is in hexadecimal format i.e. each digit represents
 :4874 four bits, so four digits represnts a full 16 bit word. The micro-
 :4875 assembler requires that a hexadecimal value that starts with a letter
 :4876 (e.g. FFFF) be preceeded by a 0 (0FFFF). So some table values will
 :4877 contain 5 hexadecimal digits. The fifth digit is not actually used.

:4878 :---

TRANSFER.DATA.LS1:

U 0070, 0000,78	COUNT.LS[0078]	:LONGWORD COUNT (NUMBER OF LONGWORDS TO TRANS-
		:FER TO LS = 120 DECIMAL) DESTINATION IS LS
U 0071, 0000,00	START.ADR[0000]	:DESTINATION START ADDRESS (START AT LS 0)
U 0072, 0000,00	WORD[0000]	:LOCATION 0
U 0073, 0000,00	WORD[0000]	
U 0074, 0000,00	WORD[0000]	:LOCATION 1
U 0075, 0000,00	WORD[0000]	
U 0076, 0000,00	WORD[0000]	:LOCATION 2
U 0077, 0000,00	WORD[0000]	
U 0078, 0000,00	WORD[0000]	:LOCATION 3
U 0079, 0000,00	WORD[0000]	
U 007A, 0000,00	WORD[0000]	:LOCATION 4
U 007B, 0000,00	WORD[0000]	
U 007C, 0000,00	WORD[0000]	:LOCATION 5
U 007D, 0000,00	WORD[0000]	
U 007E, 0000,00	WORD[0000]	:LOCATION 6
U 007F, 0000,00	WORD[0000]	
U 0080, 0000,00	WORD[0000]	:LOCATION 7

ZZ-ENKCF-1.4	:	ENKCF.MIC	LS DATA SECTION	M 9	15-MAR-83	Fiche 1 Frame M9	Sequence 116	Page 110
:	:	ENKCF.MCR	MICRO2 1M(01)	15-MAR-83	11:46:22	ENKCF Micro-diagnostic for IDC module	Rev 01.40	
:	:	ENKCF.MIC	LS DATA SECTION					

U 0081, 0000,00	:4909	WORD[0000]	
	:4910		
U 0082, 0000,00	:4911	WORD[0000]	;LOCATION 8
U 0083, 0000,00	:4912	WORD[0000]	
	:4913		
U 0084, 0000,00	:4914	WORD[0000]	;LOCATION 9
U 0085, 0000,00	:4915	WORD[0000]	
	:4916		
U 0086, 0000,00	:4917	WORD[0000]	;LOCATION 0A
U 0087, 0000,00	:4918	WORD[0000]	
	:4919		
U 0088, 0000,00	:4920	WORD[0000]	;LOCATION 0B
U 0089, 0000,00	:4921	WORD[0000]	
	:4922		
U 008A, 0000,00	:4923	WORD[0000]	;LOCATION 0C
U 008B, 0000,00	:4924	WORD[0000]	
	:4925		
U 008C, 0000,00	:4926	WORD[0000]	;LOCATION 0D
U 008D, 0000,00	:4927	WORD[0000]	
	:4928		
U 008E, 0000,00	:4929	WORD[0000]	;LOCATION 0E
U 008F, 0000,00	:4930	WORD[0000]	
	:4931		
U 0090, 0000,00	:4932	WORD[0000]	;LOCATION 0F
U 0091, 0000,00	:4933	WORD[0000]	
	:4934		
U 0092, 0000,00	:4935	WORD[0000]	;LOCATION 10
U 0093, 0000,00	:4936	WORD[0000]	
	:4937		
U 0094, 0000,00	:4938	WORD[0000]	;LOCATION 11
U 0095, 0000,00	:4939	WORD[0000]	
	:4940		
U 0096, 0000,00	:4941	WORD[0000]	;LOCATION 12
U 0097, 0000,00	:4942	WORD[0000]	
	:4943		
U 0098, 0000,FF	:4944	WORD[00FF]	;LOCATION 13
U 0099, 0000,00	:4945	WORD[0000]	
	:4946		
U 009A, 00FF,FF	:4947	WORD[0FFFF]	;LOCATION 14
U 009B, 0000,00	:4948	WORD[0000]	
	:4949		
U 009C, 0000,00	:4950	WORD[0000]	;LOCATION 15
U 009D, 00FF,00	:4951	WORD[0FF00]	
	:4952		
U 009E, 00FF,00	:4953	WORD[0FF00]	;LOCATION 16
U 009F, 00FF,FF	:4954	WORD[0FFFF]	
	:4955		
U 00A0, 003F,FF	:4956	WORD[3FFF]	;LOCATION 17
U 00A1, 0001,00	:4957	WORD[0100]	
	:4958		
U 00A2, 003F,FF	:4959	WORD[3FFF]	;LOCATION 18
U 00A3, 007F,FE	:4960	WORD[7FFE]	
	:4961		
U 00A4, 00FF,FF	:4962	WORD[0FFFF]	;LOCATION 19
U 00A5, 00FE,7F	:4963	WORD[0FE7F]	

ZZ-
:
:

U C
U C
U C

U C
U C
U C

U C
U C

U 00A6, 0000,00	:4964	WORD[0000]	:LOCATION 1A
U 00A7, 007F,F8	:4965	WORD[7FF8]	
	:4966		
U 00A8, 0080,00	:4967	WORD[8000]	:LOCATION 1B
U 00A9, 007F,FF	:4968	WORD[7FFF]	
	:4969		
U 0CAA, 0000,00	:4970	WORD[0000]	:LOCATION 1C
U 00AB, 0000,00	:4971	WORD[0000]	
	:4972		
U 00AC, 0000,00	:4973	WORD[0000]	:LOCATION 1D
U 00AD, 0000,00	:4974	WORD[0000]	
	:4975		
U 00AE, 0005,00	:4976	WORD[0500]	:LOCATION 1E
U 00AF, 0000,80	:4977	WORD[0080]	
	:4978		
U 00B0, 0005,00	:4979	WORD[0500]	:LOCATION 1F
U 00B1, 0000,00	:4980	WORD[0000]	
	:4981		
U 00B2, 0000,01	:4982	WORD[0001]	:LOCATION 20
U 00B3, 0000,00	:4983	WORD[0000]	
	:4984		
U 00B4, 0000,02	:4985	WORD[0002]	:LOCATION 21
U 00B5, 0000,00	:4986	WORD[0000]	
	:4987		
U 00B6, 0000,04	:4988	WORD[0004]	:LOCATION 22
U 00B7, 0000,00	:4989	WORD[0000]	
	:4990		
U 00B8, 0000,08	:4991	WORD[0008]	:LOCATION 23
U 00B9, 0000,00	:4992	WORD[0000]	
	:4993		
U 00BA, 0000,10	:4994	WORD[0010]	:LOCATION 24
U 00BB, 0000,00	:4995	WORD[0000]	
	:4996		
U 00BC, 0000,20	:4997	WORD[0020]	:LOCATION 25
U 00BD, 0000,00	:4998	WORD[0000]	
	:4999		
U 00BE, 0000,40	:5000	WORD[0040]	:LOCATION 26
U 00BF, 0000,00	:5001	WORD[0000]	
	:5002		
U 00C0, 0000,80	:5003	WORD[0080]	:LOCATION 27
U 00C1, 0000,00	:5004	WORD[0000]	
	:5005		
U 00C2, 0001,00	:5006	WORD[0100]	:LOCATION 28
U 00C3, 0000,00	:5007	WORD[0000]	
	:5008		
U 00C4, 0002,00	:5009	WORD[0200]	:LOCATION 29
U 00C5, 0000,00	:5010	WORD[0000]	
	:5011		
U 00C6, 0004,00	:5012	WORD[0400]	:LOCATION 2A
U 00C7, 0000,00	:5013	WORD[0000]	
	:5014		
U 00C8, 0008,00	:5015	WORD[0800]	:LOCATION 2B
U 00C9, 0000,00	:5016	WORD[0000]	
	:5017		
	:5018		

ZZ-ENKCF-1.4	:	ENKCF.MIC	LS DATA SECTION	B 10	Fiche 1	Frame B10	Sequence 118
:	:	MICRO2 1M(01)	15-MAR-83 11:46:22	15-MAR-83	ENKCF Micro-diagnostic for IDC module	Rev 01.40	Page 112
:	:	LS DATA SECTION					

U 00CA, 0010,00	:5019	WORD[1000]	;LOCATION 2C
U 00CB, 0000,00	:5020	WORD[0000]	
	:5021		
U 00CC, 0020,00	:5022	WORD[2000]	;LOCATION 2D
U 00CD, 0000,00	:5023	WORD[0000]	
	:5024		
U 00CE, 0040,00	:5025	WORD[4000]	;LOCATION 2E
U 00CF, 0000,00	:5026	WORD[0000]	
	:5027		
U 00D0, 0080,00	:5028	WORD[8000]	;LOCATION 2F
U 00D1, 0000,00	:5029	WORD[0000]	
	:5030		
U 00D2, 0000,00	:5031	WORD[0000]	;LOCATION 30
U 00D3, 0000,01	:5032	WORD[0001]	
	:5033		
U 00D4, 0000,00	:5034	WORD[0000]	;LOCATION 31
U 00D5, 0000,02	:5035	WORD[0002]	
	:5036		
U 00D6, 0000,00	:5037	WORD[0000]	;LOCATION 32
U 00D7, 0000,04	:5038	WORD[0004]	
	:5039		
U 00D8, 0000,00	:5040	WORD[0000]	;LOCATION 33
U 00D9, 0000,08	:5041	WORD[0008]	
	:5042		
U 00DA, 0000,00	:5043	WORD[0000]	;LOCATION 34
U 00DB, 0000,10	:5044	WORD[0010]	
	:5045		
U 00DC, 0000,00	:5046	WORD[0000]	;LOCATION 35
U 00DD, 0000,20	:5047	WORD[0020]	
	:5048		
U 00DE, 0000,00	:5049	WORD[0000]	;LOCATION 36
U 00DF, 0000,40	:5050	WORD[0040]	
	:5051		
U 00E0, 0000,00	:5052	WORD[0000]	;LOCATION 37
U 00E1, 0000,80	:5053	WORD[0080]	
	:5054		
U 00E2, 0000,00	:5055	WORD[0000]	;LOCATION 38
U 00E3, 0001,00	:5056	WORD[0100]	
	:5057		
U 00E4, 0000,00	:5058	WORD[0000]	;LOCATION 39
U 00E5, 0002,00	:5059	WORD[0200]	
	:5060		
U 00E6, 0000,00	:5061	WORD[0000]	;LOCATION 3A
U 00E7, 0004,00	:5062	WORD[0400]	
	:5063		
U 00E8, 0000,00	:5064	WORD[0000]	;LOCATION 3B
U 00E9, 0008,00	:5065	WORD[0800]	
	:5066		
U 00EA, 0000,00	:5067	WORD[0000]	;LOCATION 3C
U 00EB, 0010,00	:5068	WORD[1000]	
	:5069		
U 00EC, 0000,00	:5070	WORD[0000]	;LOCATION 3D
U 00ED, 0020,00	:5071	WORD[2000]	
	:5072		
U 00EE, 0000,00	:5073	WORD[0000]	;LOCATION 3E

U

U 0114, 0000,00	:5129	WORD[0000]	:LOCATION 51
U 0115, 0000,00	:5130	WORD[0000]	
	:5131		
U 0116, 0000,00	:5132	WORD[0000]	:LOCATION 52
U 0117, 0000,00	:5133	WORD[0000]	
	:5134		
U 0118, 0000,00	:5135	WORD[0000]	:LOCATION 53
U 0119, 0000,00	:5136	WORD[0000]	
	:5137		
U 011A, 0000,00	:5138	WORD[0000]	:LOCATION 54
U 011B, 0000,00	:5139	WORD[0000]	
	:5140		
U 011C, 0000,00	:5141	WORD[0000]	:LOCATION 55
U 011D, 0000,00	:5142	WORD[0000]	
	:5143		
U 011E, 0000,00	:5144	WORD[0000]	:LOCATION 56
U 011F, 0000,00	:5145	WORD[0000]	
	:5146		
U 0120, 0000,00	:5147	WORD[0000]	:LOCATION 57
U 0121, 0000,00	:5148	WORD[0000]	
	:5149		
U 0122, 0000,00	:5150	WORD[0000]	:LOCATION 58
U 0123, 0000,00	:5151	WORD[0000]	
	:5152		
U 0124, 0000,00	:5153	WORD[0000]	:LOCATION 59
U 0125, 0000,00	:5154	WORD[0000]	
	:5155		
U 0126, 00F0,00	:5156	WORD[0F000]	:LOCATION 5A
U 0127, 00FF,FF	:5157	WORD[0FFFF]	
	:5158		
U 0128, 00F0,02	:5159	WORD[0F002]	:LOCATION 5B
U 0129, 00FF,FF	:5160	WORD[0FFFF]	
	:5161		
U 012A, 00F0,04	:5162	WORD[0F004]	:LOCATION 5C
U 012B, 00FF,FF	:5163	WORD[0FFFF]	
	:5164		
U 012C, 00F0,06	:5165	WORD[0F006]	:LOCATION 5D
U 012D, 00FF,FF	:5166	WORD[0FFFF]	
	:5167		
U 012E, 00F0,08	:5168	WORD[0F008]	:LOCATION 5E
U 012F, 00FF,FF	:5169	WORD[0FFFF]	
	:5170		
U 0130, 00F0,0E	:5171	WORD[0F00E]	:LOCATION 5F
U 0131, 00FF,FF	:5172	WORD[0FFFF]	
	:5173		
U 0132, 0000,03	:5174	WORD[0003]	:LOCATION 60
U 0133, 0000,00	:5175	WORD[0000]	
	:5176		
U 0134, 0000,12	:5177	WORD[0012]	:LOCATION 61
U 0135, 0000,00	:5178	WORD[0000]	
	:5179		
U 0136, 0033,33	:5180	WORD[3333]	:LOCATION 62
U 0137, 0033,33	:5181	WORD[3333]	
	:5182		
	:5183		

U

ZZ-ENKCF-1.4	:	ENKCF.MIC	LS DATA SECTION	F 10	15-MAR-83	Fiche 1 Frame F10	Sequence 122	
:	:	ENKCF.MCR	MICRO2 1M(01)	15-MAR-83	11:46:22	ENKCF Micro-diagnostic for IDC module	Rev 01.40	Page 116
:	:	ENKCF.MIC	LS DATA SECTION					

U 015D, 0000,00	:	5239	WORD[0000]	
	:	5240		
U 015E, 0000,00	:	5241	WORD[0000]	;LOCATION 76
U 015F, 0000,00	:	5242	WORD[0000]	
	:	5243		
U 0160, 0000,00	:	5244	WORD[0000]	;LOCATION 77
U 0161, 0000,00	:	5245	WORD[0000]	
	:	5246		
	:	5247	.TOC	'PROGRAM SPECIFIC DATA SECTION (LS 80-F7)'
	:	5248		
	:	5249		This section represents the second half of LS. It is only accessible
	:	5250		with a MOV instruction, or one that uses the MOVE microinstruction for-
	:	5251		mat. This section will be loaded in a separate transfer.
	:	5252		
	:	5253	TRANSFER.DATA.LS2:	
U 0162, 0000,78	:	5254	COUNT.LS[0078]	;LONGWORD COUNT (NUMBER OF LONGWORDS TO TRANS-
	:	5255		;FER TO LS = 120 DECIMAL) DESTINATION IS LS
U 0163, 0000,80	:	5256	START.ADR[0080]	;DESTINATION START ADDRESS (START AT LS 80(H))
	:	5257		
U 0164, 0000,00	:	5258	WORD[0000]	;LOCATION 80
U 0165, 0010,00	:	5259	WORD[1000]	
	:	5260		
U 0166, 0000,02	:	5261	WORD[0002]	;LOCATION 81
U 0167, 0010,00	:	5262	WORD[1000]	
	:	5263		
U 0168, 0000,04	:	5264	WORD[0004]	;LOCATION 82
U 0169, 0010,00	:	5265	WORD[1000]	
	:	5266		
U 016A, 0000,06	:	5267	WORD[0006]	;LOCATION 83
U 016B, 0010,00	:	5268	WORD[1000]	
	:	5269		
U 016C, 0000,08	:	5270	WORD[0008]	;LOCATION 84
U 016D, 0010,00	:	5271	WORD[1000]	
	:	5272		
U 016E, 0000,0A	:	5273	WORD[000A]	;LOCATION 85
U 016F, 0010,00	:	5274	WORD[1000]	
	:	5275		
U 0170, 0000,0C	:	5276	WORD[000C]	;LOCATION 86
U 0171, 0010,00	:	5277	WORD[1000]	
	:	5278		
U 0172, 0000,0E	:	5279	WORD[000E]	;LOCATION 87
U 0173, 0010,00	:	5280	WORD[1000]	
	:	5281		
U 0174, 0000,40	:	5282	WORD[0040]	;LOCATION 88
U 0175, 0000,00	:	5283	WORD[0000]	
	:	5284		
U 0176, 0000,80	:	5285	WORD[0080]	;LOCATION 89
U 0177, 0000,00	:	5286	WORD[0000]	
	:	5287		
U 0178, 0000,00	:	5288	WORD[0000]	;LOCATION 8A
U 0179, 0000,00	:	5289	WORD[0000]	
	:	5290		
U 017A, 0001,00	:	5291	WORD[0100]	;LOCATION 8B
U 017B, 0000,00	:	5292	WORD[0000]	
	:	5293		

ZZ-ENKCF-1.4	:	ENKCF.MIC	PROGRAM SPECIFIC DATA SECTION (LS 80-F7)	G 10	Fiche 1	Frame G10	Sequence 123	Page 117
:	:	ENKCF.MIC		15-MAR-83	11:46:22	ENKCF Micro-diagnostic for IDC module	Rev 01.40	
:	:	ENKCF.MIC						

U 017C, 0002,00	:5294	WORD[0200]	:LOCATION 8C
U 017D, 0000,00	:5295	WORD[0000]	
	:5296		
U 017E, 0003,00	:5297	WORD[0300]	:LOCATION 8D
U 017F, 0000,00	:5298	WORD[0000]	
	:5299		
U 0180, 0000,00	:5300	WORD[0000]	:LOCATION 8E
U 0181, 0000,40	:5301	WORD[0040]	
	:5302		
U 0182, 0000,00	:5303	WORD[0000]	:LOCATION 8F
U 0183, 0000,80	:5304	WORD[0080]	
	:5305		
U 0184, 0000,00	:5306	WORD[0000]	:LOCATION 90
U 0185, 0002,00	:5307	WORD[0200]	
	:5308		
U 0186, 00FC,31	:5309	WORD[0FC31]	:LOCATION 91
U 0187, 00C5,3F	:5310	WORD[0C53F]	
	:5311		
U 0188, 0000,00	:5312	WORD[0000]	:LOCATION 92
U 0189, 0000,00	:5313	WORD[0000]	
	:5314		
U 018A, 0001,44	:5315	WORD[0144]	:LOCATION 93
U 018B, 0002,40	:5316	WORD[0240]	
	:5317		
U 018C, 0002,8A	:5318	WORD[028A]	:LOCATION 94
U 018D, 0000,80	:5319	WORD[0080]	
	:5320		
U 018E, 0001,86	:5321	WORD[0186]	:LOCATION 95
U 018F, 0002,80	:5322	WORD[0280]	
	:5323		
U 0190, 0000,4E	:5324	WORD[004E]	:LOCATION 96
U 0191, 0002,80	:5325	WORD[0280]	
	:5326		
U 0192, 0003,CE	:5327	WORD[03CE]	:LOCATION 97
U 0193, 0000,40	:5328	WORD[0040]	
	:5329		
U 0194, 0000,1A	:5330	WORD[001A]	:LOCATION 98
U 0195, 0000,00	:5331	WORD[0000]	
	:5332		
U 0196, 0000,00	:5333	WORD[0000]	:LOCATION 99
U 0197, 0000,00	:5334	WORD[0000]	
	:5335		
U 0198, 0000,00	:5336	WORD[0000]	:LOCATION 9A
U 0199, 0000,00	:5337	WORD[0000]	
	:5338		
U 019A, 0000,00	:5339	WORD[0000]	:LOCATION 9B
U 019B, 0000,00	:5340	WORD[0000]	
	:5341		
U 019C, 0000,00	:5342	WORD[0000]	:LOCATION 9C
U 019D, 0000,00	:5343	WORD[0000]	
	:5344		
U 019E, 0000,00	:5345	WORD[0000]	:LOCATION 9D
U 019F, 0000,00	:5346	WORD[0000]	
	:5347		
U 01A0, 0000,00	:5348	WORD[0000]	:LOCATION 9E

ZZ-ENKCF-1.4	:	ENKCF.MIC	PROGRAM SPECIFIC DATA SECTION (LS 80-F7)	H 10	15-MAR-83 11:46:22	Fiche 1 Frame H10	Sequence 124	Page 118
:	:	ENKCF.MCR	MICRO2 1M(01)			ENKCF Micro-diagnostic for IDC module	Rev 01.40	
:	:	ENKCF.MIC						

U 01A1, 0000,00	:	5349	WORD[0000]	
	:	5350		
U 01A2, 0000,00	:	5351	WORD[0000]	;LOCATION 9F
U 01A3, 0000,00	:	5352	WORD[0000]	
	:	5353		
U 01A4, 0000,00	:	5354	WORD[0000]	;LOCATION 0A0
U 01A5, 0000,00	:	5355	WORD[0000]	
	:	5356		
U 01A6, 0000,00	:	5357	WORD[0000]	;LOCATION 0A1
U 01A7, 0000,00	:	5358	WORD[0000]	
	:	5359		
U 01A8, 0000,00	:	5360	WORD[0000]	;LOCATION 0A2
U 01A9, 0000,00	:	5361	WORD[0000]	
	:	5362		
U 01AA, 0000,00	:	5363	WORD[0000]	;LOCATION 0A3
U 01AB, 0000,00	:	5364	WORD[0000]	
	:	5365		
U 01AC, 0000,00	:	5366	WORD[0000]	;LOCATION 0A4
U 01AD, 0000,00	:	5367	WORD[0000]	
	:	5368		
U 01AE, 0000,00	:	5369	WORD[0000]	;LOCATION 0A5
U 01AF, 0000,00	:	5370	WORD[0000]	
	:	5371		
U 01B0, 0000,00	:	5372	WORD[0000]	;LOCATION 0A6
U 01B1, 0000,00	:	5373	WORD[0000]	
	:	5374		
U 01B2, 0000,00	:	5375	WORD[0000]	;LOCATION 0A7
U 01B3, 0000,00	:	5376	WORD[0000]	
	:	5377		
U 01B4, 0000,00	:	5378	WORD[0000]	;LOCATION 0A8
U 01B5, 0000,00	:	5379	WORD[0000]	
	:	5380		
U 01B6, 0000,00	:	5381	WORD[0000]	;LOCATION 0A9
U 01B7, 0000,00	:	5382	WORD[0000]	
	:	5383		
U 01B8, 0000,00	:	5384	WORD[0000]	;LOCATION 0AA
U 01B9, 0000,00	:	5385	WORD[0000]	
	:	5386		
U 01BA, 0000,00	:	5387	WORD[0000]	;LOCATION 0AB
U 01BB, 0000,00	:	5388	WORD[0000]	
	:	5389		
U 01BC, 0000,00	:	5390	WORD[0000]	;LOCATION 0AC
U 01BD, 0000,00	:	5391	WORD[0000]	
	:	5392		
U 01BE, 0000,00	:	5393	WORD[0000]	;LOCATION 0AD
U 01BF, 0000,00	:	5394	WORD[0000]	
	:	5395		
U 01C0, 0000,00	:	5396	WORD[0000]	;LOCATION 0AE
U 01C1, 0000,00	:	5397	WORD[0000]	
	:	5398		
U 01C2, 0000,00	:	5399	WORD[0000]	;LOCATION 0AF
U 01C3, 0000,00	:	5400	WORD[0000]	
	:	5401		
U 01C4, 0000,00	:	5402	WORD[0000]	;LOCATION 0B0
U 01C5, 0000,00	:	5403	WORD[0000]	

ZZ
:
:

U
U
U

U 01C6, 0000,00	:5404	WORD[0000]	:LOCATION 0B1
U 01C7, 0000,00	:5405	WORD[0000]	
	:5406		
U 01C8, 0000,00	:5407	WORD[0000]	:LOCATION 0B2
U 01C9, 0000,00	:5408	WORD[0000]	
	:5409		
U 01CA, 0000,00	:5410	WORD[0000]	:LOCATION 0B3
U 01CB, 0000,00	:5411	WORD[0000]	
	:5412		
U 01CC, 0000,00	:5413	WORD[0000]	:LOCATION 0B4
U 01CD, 0000,00	:5414	WORD[0000]	
	:5415		
U 01CE, 0000,00	:5416	WORD[0000]	:LOCATION 0B5
U 01CF, 0000,00	:5417	WORD[0000]	
	:5418		
U 01D0, 0000,00	:5419	WORD[0000]	:LOCATION 0B6
U 01D1, 0000,00	:5420	WORD[0000]	
	:5421		
U 01D2, 0000,00	:5422	WORD[0000]	:LOCATION 0B7
U 01D3, 0000,00	:5423	WORD[0000]	
	:5424		
U 01D4, 0000,00	:5425	WORD[0000]	:LOCATION 0B8
U 01D5, 0000,00	:5426	WORD[0000]	
	:5427		
U 01D6, 00FF,FF	:5428	WORD[0FFFF]	:LOCATION 0B9
U 01D7, 0000,00	:5429	WORD[0000]	
	:5430		
U 01D8, 0000,86	:5431	WORD[0086]	:LOCATION 0BA
U 01D9, 0000,00	:5432	WORD[0000]	
	:5433		
U 01DA, 0000,00	:5434	WORD[0000]	:LOCATION 0BB
U 01DB, 00FF,F8	:5435	WORD[0FFF8]	
	:5436		
U 01DC, 0002,C4	:5437	WORD[02C4]	:LOCATION 0BC
U 01DD, 0028,80	:5438	WORD[2880]	
	:5439		
U 01DE, 0001,8A	:5440	WORD[018A]	:LOCATION 0BD
U 01DF, 0012,40	:5441	WORD[1240]	
	:5442		
U 01E0, 0003,86	:5443	WORD[0386]	:LOCATION 0BE
U 01E1, 000A,00	:5444	WORD[0A00]	
	:5445		
U 01E2, 0000,CE	:5446	WORD[00CE]	:LOCATION 0BF
U 01E3, 003A,00	:5447	WORD[3A00]	
	:5448		
U 01E4, 0003,CE	:5449	WORD[03CE]	:LOCATION 0C0
U 01E5, 0000,40	:5450	WORD[0040]	
	:5451		
U 01E6, 00FF,7F	:5452	WORD[0FF7F]	:LOCATION 0C1
U 01E7, 00FF,FF	:5453	WORD[0FFFF]	
	:5454		
U 01E8, 00FF,FF	:5455	WORD[0FFFF]	:LOCATION 0C2
U 01E9, 00FF,BF	:5456	WORD[0FFBF]	
	:5457		
	:5458		

J 10
15-MAR-83 11:46:22

ZZ-ENKCF-1.4 : ENKCF.MIC
: ENKCF.MCR
: ENKCF.MIC

Fiche 1 Frame J10
ENKCF Micro-diagnostic for IDC module
Sequence 126
Rev 01.40
Page 120

PROGRAM SPECIFIC DATA SECTION (LS 80-F7)

U 01EA, 00FF,FF	:5459	WORD[0FFFF]	:LOCATION 0C3
U 01EB, 00FF,7F	:5460	WORD[0FF7F]	
	:5461		
U 01EC, 0000,03	:5462	WORD[0003]	:LOCATION 0C4
U 01ED, 0000,00	:5463	WORD[0000]	
	:5464		
U 01EE, 0000,FE	:5465	WORD[00FE]	:LOCATION 0C5
U 01EF, 0000,00	:5466	WORD[0000]	
	:5467		
U 01F0, 0001,FE	:5468	WORD[01FE]	:LOCATION 0C6
U 01F1, 0000,00	:5469	WORD[0000]	
	:5470		
U 01F2, 0000,00	:5471	WORD[0000]	:LOCATION 0C7
U 01F3, 0000,00	:5472	WORD[0000]	
	:5473		
U 01F4, 00FF,F0	:5474	WORD[0FFF0]	:LOCATION 0C8
U 01F5, 00FF,FF	:5475	WORD[0FFFF]	
	:5476		
U 01F6, 0000,06	:5477	WORD[0006]	:LOCATION 0C9
U 01F7, 0000,00	:5478	WORD[0000]	
	:5479		
U 01F8, 00FF,FF	:5480	WORD[0FFFF]	:LOCATION 0CA
U 01F9, 0000,07	:5481	WORD[0007]	
	:5482		
U 01FA, 0000,0F	:5483	WORD[000F]	:LOCATION 0CB
U 01FB, 0000,00	:5484	WORD[0000]	
	:5485		
U 01FC, 00FC,FF	:5486	WORD[0FCFF]	:LOCATION 0CC
U 01FD, 00FF,FF	:5487	WORD[0FFFF]	
	:5488		
U 01FE, 00FF,FC	:5489	WORD[0FFFC]	:LOCATION 0CD
U 01FF, 00FF,FF	:5490	WORD[0FFFF]	
	:5491		
U 0200, 00FF,80	:5492	WORD[0FF80]	:LOCATION 0CE
U 0201, 00FF,FF	:5493	WORD[0FFFF]	
	:5494		
U 0202, 0000,00	:5495	WORD[0000]	:LOCATION 0CF
U 0203, 0020,00	:5496	WORD[2000]	
	:5497		
U 0204, 0005,00	:5498	WORD[0500]	:LOCATION 0D0
U 0205, 0020,10	:5499	WORD[2010]	
	:5500		
U 0206, 0005,00	:5501	WORD[0500]	:LOCATION 0D1
U 0207, 00A0,10	:5502	WORD[0A010]	
	:5503		
U 0208, 00FF,FF	:5504	WORD[0FFFF]	:LOCATION 0D2
U 0209, 00FF,CF	:5505	WORD[0FFCF]	
	:5506		
U 020A, 00E0,00	:5507	WORD[0E000]	:LOCATION 0D3
U 020B, 00FF,FF	:5508	WORD[0FFFF]	
	:5509		
U 020C, 0000,69	:5510	WORD[0069]	:LOCATION 0D4
U 020D, 0000,00	:5511	WORD[0000]	
	:5512		
U 020E, 0068,37	:5513	WORD[6837]	:LOCATION 0D5

U 4
U 4

U

U

U

U

U

U

U

10

U

22

22

U

U
U
U
U

```

:5619 .PAGE 'WCS SUBROUTINES FOR CONSOLE USE'
:5620 .TOC  ' GENERATE TRANSFER DATA'
:5621 :
:5622 This routine is used by the diagnostic monitor to load WR 0 with a
:5623 data pattern from the console. The monitor will set CONSOLE ATTN if
:5624 a 1 is to be generated, or clear CONSOLE ATTN if a 0 is to be gen-
:5625 erated. It will then single step this routine to shift the data bit
:5626 into WR 0. This routine loads a 32 bit value much more quickly than
:5627 shifting each instruction into the CSR from the monitor, and is used
:5628 mainly to set up data to load in LS for constants.
:5629 :
:5630 :
:5631 ***WARNING***
:5632 :
:5633 This routine must start at 600 and end at 605. DO NOT move this
:5634 routine to any other area!!
:5635 :
:5636 :
:5637 :
:5638 GEN.XFER.DATA:
U 0600, 2F80,15 :5639 CLR WR[0] ;CLEAR A WORKING REG
U 0601, A0C0,15 :5640 COM WR[0] ;NOW WR 0 IS ALL ONES
:5641 :
:5642 LOOP.XFER:
U 0602, 5B00,18 :5642 SKIP.IF[CONSOLE.ATTN] ;SKIP NEXT INSTRUCTION IS CONSOLE ATTN
:5643 : ; IS SET (GENERATING A 1)
:5644 ASHL WR[0], ;SHIFT A 0 INTO BIT 0 OF WR 0
:5645 SKIP ; AND SKIP ROL
U 0603, A3D0,1E :5645 ROL WR[0] ;SHIFT A 1 INTO BIT 0 OF WR 0
U 0604, A3C0,15 :5646 ROL WR[0] ;SHIFT A 1 INTO BIT 0 OF WR 0
U 0605, 0860,24 :5647 JMP [LOOP.XFER] ;REPEAT
:5648 :
:5649 :
:5650 :
:5651 DIAGNOSTIC VERSION NUMBER IS STORED HERE
:5652 :
:5653 :
:5654 ***WARNING***
:5655 :
:5656 These words must be at location 606 and 607. DO NOT move this word
:5657 to any other area!!
:5658 :
:5659 :
:5660 :
U 0606, 0001,40 :5661 WORD[0140] ;VERSION 01.40
U 0607, 0043,46 :5662 ASCII [CF] ;LAST 2 LETTERS OF FILE NAME [ENKCF]
:5663 :
:5664 :

```

U
U
U
U

:5665 :PAGE " DEPOSIT MCT CSR ROUTINE"
:5666 :+++
:5667 :
:5668 :FUNCTIONAL DESCRIPTION:
:5669 :
:5670 :The deposit to CSR routine is used to write the memory controller's
:5671 :control and status register. The memory controller has three CSRs
:5672 :named CSR 0, CSR 1, and CSR 2.
:5673 :CSR 0 contains checkbits or syndrome bits returned from the ECC logic
:5674 :after certain diagnostic routines. It is NOT writable directly with
:5675 :this routine.
:5676 :CSR 1 contains error bits set if an error occurs on a memory access.
:5677 :It does contain five maintenance bits (bits 29-25) which are writable.
:5678 :There are also seven checkbits (bits 6-0) which are writable, but not
:5679 :readable. These bits are used to write checkbits into the ECC chips.
:5680 :CSR 2 contains error bits set if a UNIBUS error occurs on a UNIBUS
:5681 :access. These are read only bits and are NOT writable by this routine.
:5682 :Therefore, CSR 1 is the only writable CSR, and only bits 29-25 can be
:5683 :both written and read back. This routine will thus force CSR 1 on a
:5684 :deposit to CSR.
:5685 :This routine will write LS 5 with data to access CSR 1, then write
:5686 :the data residing in LS 6 into the CSR. It will then issue CPU ATTN
:5687 :to inform the console that it has completed the function.
:5688 :NOTE: The error bits of CSR 1 are cleared on any write to CSR 1.
:5689 :These are bits 31, 30 and 23-14. Bits 13-0 and bit 24 are unused.
:5690 :
:5691 :CALLING SEQUENCE:
:5692 :
:5693 :The console loads the address of a pointer to this routine (a JMP in-
:5694 :struction at WCS location 50) into the UPC and starts the CPU.
:5695 :
:5696 :INPUT PARAMETERS:
:5697 :
:5698 :LS location 6 must have been written by the console with the desired
:5699 :data to be deposited in CSR 1 previous to executing this routine.
:5700 :
:5701 :IMPLICIT INPUTS:
:5702 :
:5703 :None
:5704 :
:5705 :OUTPUT PARAMATERS:
:5706 :
:5707 :CSR 1 bits 29-25 and 6-0 are written with data from LS 6.
:5708 :
:5709 :IMPLICIT OUTPUTS:
:5710 :
:5711 :None
:5712 :
:5713 :SIDE EFFECTS:
:5714 :
:5715 :WR 0 will be modified by this routine.
:5716 :LS 5 will be modified by this routine.
:5717 :CSR 1 bits 31, 30 and 23-14 are cleared.
:5718 :
:5719 :---

U (

U (

U (

```

:5720
:5721 DEPOSIT.CSR:
:5722
U 0608, 3644,15 :5723     MOV LS[CSR1] TO WR[0]           ;SET BIT 2 IN WR 0 AS CSR NUMBER TO
:5724     ;WRITE (BITS 2 AND 3 ARE CSR NUMBER)
U 0609, BE0A,15 :5725     MOV WR[0] TO LS[T5.SUB]       ;PUT CSR NUMBER IN LS LOCATION 5 (CSR
:5726     ;1)
U 060A, 1D0B,F5 :5727     MEM.REQ[WRITE.CSR] ADRS[T5.SUB] DT[LONG] ;ISSUE MEMORY REQUEST TO
:5728     ;ACCESS CSR 1
U 060B, B20C,15 :5729     WRITE.MEM LS[T6.SUB]       ;SEND DATA IN LS LOCATION 6 TO CSR 1
U 060C, 8868,74 :5730     JMP [WAIT.SUB]           ;JUMP TO SET ATTN AND WAIT
  
```

ZZ-ENKCF-1.4 : ENKCF.MIC EXAMINE MCT CSR ROUTINE C 11
 : ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Sequence 132
 : ENKCF.MIC EXAMINE MCT CSR ROUTINE Rev 01.40 Page 126

```

:5731 .PAGE  " EXAMINE MCT CSR ROUTINE"
:5732 .+++
:5733
:5734 .FUNCTIONAL DESCRIPTION:
:5735
:5736 .The examine CSR routine reads the memory controller's control and
:5737 .status register. There are 3 CSRs labeled CSR 0, CSR 1 and CSR 2.
:5738 .CSR 0 contains checkbits or syndrome bits from the ECC logic.
:5739 .These are contained in bits 6-0 of the CSR. Other bits are UNPRE-
:5740 .DICTABLE. Bits 6-0 are only written by special diagnostic routines
:5741 .(they are not writable by the DEPOSIT CSR command).
:5742 .CSR 1 contains error bits and maintenance bits. The error bits are
:5743 .31, 30, and 23-14. Maintenance bits are 29-25. The other bits are
:5744 .UNPREDICTABLE. The error bits are written during memory functions
:5745 .only and are not writable via the DEPOSIT CSR command. The maintenance
:5746 .bits are writable. Note that a DEPOSIT CSR command will write the
:5747 .maintenance bits and clear all error bits.
:5748 .CSR 2 contains three bits (bits 16-14) which are readable. These are
:5749 .set when a UNIBUS error occurs on a UNIBUS access. They are not
:5750 .directly writable with the DEPOSIT CSR command.
:5751 .The console loads LS 5 with the CSR number (0, 1, or 2) to be read.
:5752 .This routine rotates LS 5 two places left to position it correctly.
:5753 .It then executes a read CSR and places the data in LS 6 for the console
:5754 .to read.
:5755
:5756 .CALLING SEQUENCE:
:5757
:5758 .The console loads the address of a pointer to this routine (a JMP in-
:5759 .struction at WCS location 51) into the UPC and starts the CPU.
:5760
:5761 .INPUT PARAMETERS:
:5762
:5763 .LS 5 contains the CSR number to be read.
:5764
:5765 .IMPLICIT INPUTS:
:5766
:5767 .None
:5768
:5769 .OUTPUT PARAMATERS:
:5770
:5771 .LS 6 - Data read from CSR selected.
:5772
:5773 .IMPLICIT OUTPUTS:
:5774
:5775 .None
:5776
:5777 .SIDE EFFECTS:
:5778
:5779 .LS 5 is rotated 2 places left.
:5780
:5781 .---
:5782
:5783 .EXAMINE.CSR:
:5784
:5785 .MOV LS[T5.SUB] TO WR[0] ;GET CSR NUMBER FROM LS 5
  
```

U 060D, 360A,15

D 11
15-MAR-83
Fiche 1 Frame D11
Sequence 133
Page 127

```

ZZ-ENKCF-1.4 : ENKCF.MIC EXAMINE MCT CSR ROUTINE
: ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40
: ENKCF.MIC EXAMINE MCT CSR ROUTINE

U 060E, A2D0,15 :5786 SHL2 WR[0] ;SHIFT NUMBER 2 PLACES LEFT TO GET
:5787 ; CSR NUMBER IN BITS 2 AND 3
U 060F, BE0A,15 :5788 MOV WR[0] TO LS[T5.SUB] ;PUT SHIFTED NUMBER IN LS 5
:5789
U 0610, 9D0B,75 :5790 MEM.REQ[READ.CSR] ADRS[T5.SUB] DT[LONG] ;ISSUE MEMORY REQUEST TO
:5791 ; ACCESS CSR SPECIFIED
U 0611, 3022,15 :5792 MOV MEM.DATA TO WR[0] ;READ CSR SELECTED
:5793
:5794 CHECK.CSR2:
U 0612, B60A,95 :5795 MOV LS[T5.SUB] TO WR[1] ;GET SHIFTED CSR NUMBER
:5796 CMP LS[#8] WITH WR[1], ;SEE IF CSR 2 WAS SELECTED
U 0613, CE46,B5 :5797 DT(LONG)&SET.ALU.CC ; SET UP CONDITION CODES
U 0614, 0861,61 :5798 JMP.IF[NEQ] TO [CHECK.CSR1] ;JUMP IF CSR 2 WAS NOT SELECTED
U 0615, C530,15 :5799 BIC LS[CSR2.MASK] TO WR[0] ;CLEAR UNUSED (UNPREDICTABLE) BITS IN
:5800 ; CSR 2.
:5801 CHECK.CSR1:
:5802 CMP LS[#4] WITH WR[1], ;SEE IF CSR 1 WAS SELECTED
U 0616, 4E44,B5 :5803 DT(LONG)&SET.ALU.CC ; SET UP CONDITION CODES
U 0617, 0861,91 :5804 JMP.IF[NEQ] TO [CHECK.CSR0] ;JUMP IF CSR 1 WAS NOT SELECTED
U 0618, C52E,15 :5805 BIC LS[CSR1.MASK] TO WR[0] ;CLEAR UNUSED (UNPREDICTABLE) BITS IN
:5806 ; CSR 1.
:5807 CHECK.CSR0:
:5808 CMP LS[#0] WITH WR[1], ;SEE IF CSR 0 WAS SELECTED
U 0619, 4E9C,B5 :5809 DT(LONG)&SET.ALU.CC ; SET UP CONDITION CODES
U 061A, 8861,D1 :5810 JMP.IF[NEQ] TO [LOAD.LS6] ;JUMP IF CSR 0 WAS NOT SELECTED
U 061B, 452C,15 :5811 BIC LS[CSR0.MASK] TO WR[0] ;CLEAR UNUSED (UNPREDICTABLE) BITS IN
:5812 ; CSR 0.
U 061C, C54E,15 :5813 BIC LS[BIT7] TO WR[0] ;BIT 7 IS ALSO UNUSED
:5814
:5815 LOAD.LS6:
U 061D, BE0C,15 :5816 MOV WR[0] TO LS[T6.SUB] ;LOAD CSR DATA INTO LS 6 FOR PRINTOUT
U 061E, 8868,74 :5817 JMP [WAIT.SUB] ;ISSUE CPU ATTN AND WAIT FOR RESPONSE

```

:5818 .page " DEPOSIT MCT TRANSLATION BUFFER ROUTINE"
:5819 :+++

:5820 :
:5821 : FUNCTIONAL DESCRIPTION:
:5822 :

:5823 : This routine allows a write to the translation buffer portion of the
:5824 : memory controller. The buffer area contains 128 decimal locations
:5825 : addressed from 0-7F(H). The remainder of the TB RAM is either unused
:5826 : or contains the UNIBUS MAP. The DEPOSIT UB command is used to write
:5827 : the UNIBUS map portion of the TB RAMs. The address specified with the
:5828 : DEPOSIT command will be the actual RAM address accessed. This routine
:5829 : will do any shifting necessary on the address specified to position it
:5830 : correctly. The address specified has been loaded by the console in LS
:5831 : location 5 prior to executing this routine. The address must be in HEX
:5832 : format.

:5833 : The data specified has been placed in LS 6 by the console prior to
:5834 : executing this routine. Bits 07-01 will be the Valid, Prot, Modify,
:5835 : and Byte Offset bits (there are four Prot bits). Bits 22-08 will
:5836 : contain the PA bits from PA 23 through PA 09 respectively. Bits 31
:5837 : through 23 and bit 0 are unused. This routine will do any shifting nec-
:5838 : essary to write the bits in TB as described. The data specified must
:5839 : be in HEX format.

:5840 :
:5841 : CALLING SEQUENCE:
:5842 :

:5843 : The console loads the address of a pointer to this routine (a JMP in-
:5844 : struction at WCS location 52) into the UPC and starts the CPU.

:5845 :
:5846 : INPUT PARAMETERS:
:5847 :

:5848 : LS 5 - TB address to be written (range 0-7F)
:5849 : LS 6 - TB data to write (22 bits, from 1-22). Bits 0 and 23-31 are
:5850 : ignored.

:5851 :
:5852 : IMPLICIT INPUTS:
:5853 :

:5854 : None

:5855 :
:5856 : OUTPUT PARAMATERS:
:5857 :

:5858 : TB will be written with specified data at specified address

:5859 :
:5860 : IMPLICIT OUTPUTS:
:5861 :

:5862 : None

:5863 :
:5864 : SIDE EFFECTS:
:5865 :

:5866 : WR 0 will be modified
:5867 : WR 1 will be modified

:5868 :
:5869 : ---

:5870 :
:5871 : DEPOSIT.TB:

:5872 : JSR [SHIFT.TB.ADR]

: GO TO SUBROUTINE TO POSITION TB AD-

```

U 0620, 360C,15 :5873      MOV LS[T6.SUB] TO WR[0]      ; DRESS CORRECTLY
U 0621, 4526,15 :5874      BIC LS[0FF] TO WR[0]      ; GET DATA FROM LS 6
:5875      :5876      :5877      :5878      :5879      :5880      :5881      :5882      :5883      :5884      :5885      :5886      :5887      :5888      :5889      :5890      :5891      :5892      :5893      :5894      :5895      :5896      :5897      :5898      :5899      :5900      :5901      :5902      :5903      :5904      :5905      :5906      :5907      :5908      :5909      :5910      :5911      :5912      :5913      :5914      :5915      :5916      :5917      :5918      :5919      :5920      :5921      :5922      :5923      :5924      :5925      :5926      :5927
U 0622, 8862,AC :5878      JSR [SHIFT.LOOP]      ; GO TO SUBROUTINE TO SHIFT WR 0 EIGHT
:5879      :5880      :5881      :5882      :5883      :5884      :5885      :5886      :5887      :5888      :5889      :5890      :5891      :5892      :5893      :5894      :5895      :5896      :5897      :5898      :5899      :5900      :5901      :5902      :5903      :5904      :5905      :5906      :5907      :5908      :5909      :5910      :5911      :5912      :5913      :5914      :5915      :5916      :5917      :5918      :5919      :5920      :5921      :5922      :5923      :5924      :5925      :5926      :5927
U 0623, E40C,15 :5880      SWAP LS[T6.SUB] WITH WR[0]      ; SWAP ORIGINAL DATA SPECIFIED WITH MOD-
:5881      :5882      :5883      :5884      :5885      :5886      :5887      :5888      :5889      :5890      :5891      :5892      :5893      :5894      :5895      :5896      :5897      :5898      :5899      :5900      :5901      :5902      :5903      :5904      :5905      :5906      :5907      :5908      :5909      :5910      :5911      :5912      :5913      :5914      :5915      :5916      :5917      :5918      :5919      :5920      :5921      :5922      :5923      :5924      :5925      :5926      :5927
U 0624, 452C,15 :5884      BIC LS[FFFFFF00] TO WR[0]      ; CLEAR ALL BUT LOW BYTE OF ORIGINAL
:5885      :5886      :5887      :5888      :5889      :5890      :5891      :5892      :5893      :5894      :5895      :5896      :5897      :5898      :5899      :5900      :5901      :5902      :5903      :5904      :5905      :5906      :5907      :5908      :5909      :5910      :5911      :5912      :5913      :5914      :5915      :5916      :5917      :5918      :5919      :5920      :5921      :5922      :5923      :5924      :5925      :5926      :5927
U 0625, 8862,AC :5886      JSR [SHIFT.LOOP]      ; GO TO SUBROUTINE TO SHIFT WR 0 EIGHT
:5887      :5888      :5889      :5890      :5891      :5892      :5893      :5894      :5895      :5896      :5897      :5898      :5899      :5900      :5901      :5902      :5903      :5904      :5905      :5906      :5907      :5908      :5909      :5910      :5911      :5912      :5913      :5914      :5915      :5916      :5917      :5918      :5919      :5920      :5921      :5922      :5923      :5924      :5925      :5926      :5927
U 0626, ED0C,15 :5890      BIS WR[0] TO LS[T6.SUB]      ; NOW LS 6 CONTAINS DATA TO WRITE IN
:5891      :5892      :5893      :5894      :5895      :5896      :5897      :5898      :5899      :5900      :5901      :5902      :5903      :5904      :5905      :5906      :5907      :5908      :5909      :5910      :5911      :5912      :5913      :5914      :5915      :5916      :5917      :5918      :5919      :5920      :5921      :5922      :5923      :5924      :5925      :5926      :5927
U 0627, 980A,F5 :5896      MEM.REQ[WRITE.TB] ADRS[T5.SUB] DT[LONG] ;ISSUE MEMORY REQUEST TO WRITE
:5897      :5898      :5899      :5900      :5901      :5902      :5903      :5904      :5905      :5906      :5907      :5908      :5909      :5910      :5911      :5912      :5913      :5914      :5915      :5916      :5917      :5918      :5919      :5920      :5921      :5922      :5923      :5924      :5925      :5926      :5927
U 0628, B20C,15 :5898      WRITE.MEM LS[T6.SUB]      ; WRITE DATA FROM LS 6 IN TB
U 0629, 8868,74 :5899      JMP [WAIT.SUB]      ; JUMP TO SET ATTN AND WAIT
:5900      :5901      :5902      :5903      :5904      :5905      :5906      :5907      :5908      :5909      :5910      :5911      :5912      :5913      :5914      :5915      :5916      :5917      :5918      :5919      :5920      :5921      :5922      :5923      :5924      :5925      :5926      :5927
:5901      :5902      :5903      :5904      :5905      :5906      :5907      :5908      :5909      :5910      :5911      :5912      :5913      :5914      :5915      :5916      :5917      :5918      :5919      :5920      :5921      :5922      :5923      :5924      :5925      :5926      :5927
:5902      :5903      :5904      :5905      :5906      :5907      :5908      :5909      :5910      :5911      :5912      :5913      :5914      :5915      :5916      :5917      :5918      :5919      :5920      :5921      :5922      :5923      :5924      :5925      :5926      :5927
:5903      :5904      :5905      :5906      :5907      :5908      :5909      :5910      :5911      :5912      :5913      :5914      :5915      :5916      :5917      :5918      :5919      :5920      :5921      :5922      :5923      :5924      :5925      :5926      :5927
:5904      :5905      :5906      :5907      :5908      :5909      :5910      :5911      :5912      :5913      :5914      :5915      :5916      :5917      :5918      :5919      :5920      :5921      :5922      :5923      :5924      :5925      :5926      :5927
U 062A, 3646,95 :5905      SHIFT.LOOP:      ; SUBROUTINE TO SHIFT WR 0 TO THE RIGHT 8 PLACES
:5906      :5907      :5908      :5909      :5910      :5911      :5912      :5913      :5914      :5915      :5916      :5917      :5918      :5919      :5920      :5921      :5922      :5923      :5924      :5925      :5926      :5927
U 062B, 2340,15 :5907      SHIFT.LOOP.1:      ; COUNTER IN WR 1 TO SHIFT 8 PLACES
:5908      :5909      :5910      :5911      :5912      :5913      :5914      :5915      :5916      :5917      :5918      :5919      :5920      :5921      :5922      :5923      :5924      :5925      :5926      :5927
U 062C, A102,B5 :5909      ROR WR[0]      ; SHIFT WR 0 ONE PLACE RIGHT
:5910      :5911      :5912      :5913      :5914      :5915      :5916      :5917      :5918      :5919      :5920      :5921      :5922      :5923      :5924      :5925      :5926      :5927
U 062D, 8862,B1 :5910      DEC WR[1]      ; DECREMENT COUNTER
:5911      :5912      :5913      :5914      :5915      :5916      :5917      :5918      :5919      :5920      :5921      :5922      :5923      :5924      :5925      :5926      :5927
U 062E, 5B00,14 :5911      DT[LONG]&SET.ALU.CC      ; AND SET CONDITION CODES
:5912      :5913      :5914      :5915      :5916      :5917      :5918      :5919      :5920      :5921      :5922      :5923      :5924      :5925      :5926      :5927
:5912      :5913      :5914      :5915      :5916      :5917      :5918      :5919      :5920      :5921      :5922      :5923      :5924      :5925      :5926      :5927
:5913      :5914      :5915      :5916      :5917      :5918      :5919      :5920      :5921      :5922      :5923      :5924      :5925      :5926      :5927
:5914      :5915      :5916      :5917      :5918      :5919      :5920      :5921      :5922      :5923      :5924      :5925      :5926      :5927
:5915      :5916      :5917      :5918      :5919      :5920      :5921      :5922      :5923      :5924      :5925      :5926      :5927
:5916      :5917      :5918      :5919      :5920      :5921      :5922      :5923      :5924      :5925      :5926      :5927
:5917      :5918      :5919      :5920      :5921      :5922      :5923      :5924      :5925      :5926      :5927
:5918      :5919      :5920      :5921      :5922      :5923      :5924      :5925      :5926      :5927
:5919      :5920      :5921      :5922      :5923      :5924      :5925      :5926      :5927
:5920      :5921      :5922      :5923      :5924      :5925      :5926      :5927
:5921      :5922      :5923      :5924      :5925      :5926      :5927
:5922      :5923      :5924      :5925      :5926      :5927
:5923      :5924      :5925      :5926      :5927
:5924      :5925      :5926      :5927
:5925      :5926      :5927
:5926      :5927
:5927

```


U 0637, 5B00,09	:5928	SKIP.IF[EQL]	;SKIP NEXT INSTRUCTION IF MSB IS 0
U 0638, 477E,15	:5929	BIS LS[BIT31] TO WR[0]	;ELSE SET BIT 31 FOR MSB
	:5930		
	:5931	MOV WR[0] TO LS[T5.SUB],	;NOW TB ADDRESS IS IN CORRECT FORM IN
	:5932		; LS LOCATION 5
U 0639, 3E0A,14	:5933	RETURN	; THEN RETURN TO MAIN CODE

```

:5934 .PAGE  " EXAMINE TRANSLATION BUFFER ROUTINE"
:5935 .+++
:5936
:5937 .FUNCTIONAL DESCRIPTION:
:5938
:5939     This routine will read the translation buffer of the memory controller.
:5940     The TB will be returned with the byte offset, modify, prot A through
:5941     prot D, and valid in bits 01-07 respectively.  PA bits 09-23 are re-
:5942     turned in bits 08-22 respectively.  Bits 23-31 and bit 0 are not used
:5943     and so will be cleared before printing the result.  The result will be
:5944     put in LS 6 for the console to print.
:5945     The routine will issue a memory request with memory function=READ.TB
:5946     and data type=LONGWORD.  The console will load LS 5 with the address
:5947     specified before executing this routine.  The routine will shift the
:5948     address around as required to address the TB location specified.  The
:5949     address specified must be in HEX format.
:5950     The TB consists of 128 decimal locations (addresses from 0-7F).  The
:5951     remaining locations in the TB RAM are either unused or used by the
:5952     UNIBUS map.  UNIBUS map addresses are read via the EXAMINE UB routine.
:5953
:5954 .CALLING SEQUENCE:
:5955
:5956     The console loads the address of a pointer to this routine (a JMP in-
:5957     struction at WCS location 53) into the UPC and starts the CPU.
:5958
:5959 .INPUT PARAMETERS:
:5960
:5961     LS 5 - TB address (0-7F) in hex format
:5962
:5963 .IMPLICIT INPUTS:
:5964
:5965     None
:5966
:5967 .OUTPUT PARAMATERS:
:5968
:5969     LS 6 - TB data from address specified
:5970
:5971 .IMPLICIT OUTPUTS:
:5972
:5973     None
:5974
:5975 .SIDE EFFECTS:
:5976
:5977     LS 5 is modified
:5978     WR 0 is modified
:5979
:5980 .---
:5981
:5982 .EXAMINE.TB:
:5983     JSR [SHIFT.TB.ADR] ;SHIFT ADDRESS SPECIFIED FROM LS 5 INTO
:5984     ; CORRECT POSITION AND REPLACE IN LS 5
:5985     MEM.REQ[READ.TB] ADRS[T5.SUB] DT[LONG] ;ISSUE MEMORY REQUEST TO READ
:5986     ; THE TB AT ADDRESS CONTAINED IN LS 5
:5987     MOV MEM.DATA TO WR[0] ;PLACE RESULT IN WR 0
:5988
  
```

U 063A, 8862,FC
 U 063B, 9C0B, F5
 U 063C, 3022,15

U 063D, 452A,15 ;5989	BIC LS[0FF00000] TO WR[0]	;CLEAR UPPER BYTE OF RESULT (UPPER BYTE
:5990		; IS NOT PART OF TB AND IS UNPREDICT-
:5991		; ABLE)
U 063E, 456E,15 ;5992	BIC LS[BIT23] TO WR[0]	;DITTO FOR BIT 23
U 063F, 4540,15 ;5993	BIC LS[BIT0] TO WR[0]	;DITTO FOR BIT 0. WR 0 NOW CONTAINS
:5994		; RESULT TO PRINT
U 0640, BE0C,15 ;5995	MOV WR[0] TO LS[T6.SUB]	;RESULT NOW IN LS 6
U 0641, 8868,74 ;5996	JMP [WAIT.SUB]	;JUMP TO SET ATTN AND WAIT

CCCCC

```
:5997 .PAGE  " DEPOSIT UNIBUS MAP ROUTINE"
:5998 .+++
:5999
:6000 .FUNCTIONAL DESCRIPTION:
:6001
:6002 .      This routine allows a write to the UNIBUS map portion of the TB on the
:6003 .      memory controller.  The map area contains 512 decimal locations
:6004 .      addressed from 200-3FF.  The remainder of the TB RAM is either unused
:6005 .      or contains the Translation Buffer information.  The DEPOSIT TB command
:6006 .      is used to write the TB portion of the TB RAMs.  The address specified
:6007 .      with the DEPOSIT command will be the actual RAM address accessed.  This
:6008 .      routine will do any shifting necessary on the address specified to pos-
:6009 .      ition it correctly.  The address specified has been loaded by the con-
:6010 .      sole in LS location 5 prior to executing this routine.  The address
:6011 .      must be in HEX format.
:6012 .      The data specified has been placed in LS 6 by the console prior to
:6013 .      executing this routine.  Bits 07-01 will be the Valid, Prot, Modify,
:6014 .      and Byte Offset bits (there are four Prot bits).  Bits 22-08 will
:6015 .      contain the PA bits from PA 23 through PA 09 respectively.  Bits 31
:6016 .      through 23 and bit 0 are unused.  This routine will do any shifting nec-
:6017 .      essary to write the bits in TB as described.  The data specified must
:6018 .      be in HEX format.
:6019
:6020 .CALLING SEQUENCE:
:6021
:6022 .      The console loads the address of a pointer to this routine (a JMP in-
:6023 .      struction at WCS location 58) into the UPC and starts the CPU.
:6024
:6025 .INPUT PARAMETERS:
:6026
:6027 .      LS 5 - UNIBUS Map address in TB (must be in range 200-3FF HEX).
:6028 .      LS 6 - Contains the data to write into the UNIBUS Map in bits 1-22.
:6029
:6030 .IMPLICIT INPUTS:
:6031
:6032 .      None
:6033
:6034 .OUTPUT PARAMATERS:
:6035
:6036 .      The UNIBUS map portion of the TB RAMs gets written with the data from
:6037 .      LS 6 at the address specified in LS 5.
:6038
:6039 .IMPLICIT OUTPUTS:
:6040
:6041 .      None
:6042
:6043 .SIDE EFFECTS:
:6044
:6045 .      WR 0 will be modified
:6046 .      WR 1 will be modified
:6047
:6048 .---
:6049
:6050 .DEPOSIT.UBS:
:6051 .      JSR [SHIFT.UB.ADR]                ;GO TO SUBROUTINE TO ARRANGE UNIBUS
```

```

U 0643, 360C,15 :6052
U 0644, 4526,15 :6053
:6054
:6055
:6056
U 0645, 8862,AC :6057
:6058
U 0646, E40C,15 :6059
:6060
:6061
:6062
U 0647, 452C,15 :6063
:6064
U 0648, 8862,AC :6065
:6066
:6067
:6068
U 0649, EDOC,15 :6069
:6070
:6071
:6072
:6073
:6074
U 064A, 1B0B,F5 :6075
:6076
U 064B, B20C,15 :6077
U 064C, 8868,74 :6078
:6079
:6080
:6081
:6082
:6083
U 064D, 360A,15 :6084
U 064E, A2D0,15 :6085
U 064F, A2D0,15 :6086
U 0650, A2D0,15 :6087
U 0651, A2D0,15 :6088
U 0652, 23D0,15 :6089
:6090
:6091
U 0653, 3E0A,14 :6092
:6093
:6094
    
```

MOV LS[T6.SUB] TO WR[0]
 BIC LS[#FF] TO WR[0]
 JSR [SHIFT.LOOP]
 SWAP LS[T6.SUB] WITH WR[0]
 BIC LS[#FFFFFF00] TO WR[0]
 JSR [SHIFT.LOOP]
 BIS WR[0] TO LS[T6.SUB]
 MEM.REQ[WRITE.UBS.MAP] ADRS[T5.SUB] DT[LONG] ;ISSUE MEMORY REQUEST TO
 WRITE.MEM LS[T6.SUB] ;WRITE THE UNIBUS MAP
 JMP [WAIT.SUB] ;JUMP TO SET ATTN AND WAIT
 :
 : SUBROUTINE TO POSITION ADDRESS SPECIFIED CORRECTLY TO ADDRESS THE
 : UNIBUS MAP.
 :
 : SHIFT.UB.ADR:
 MOV LS[T5.SUB] TO WR[0]
 SHL2 WR[0]
 SHL2 WR[0]
 SHL2 WR[0]
 SHL2 WR[0]
 SHL2 WR[0]
 ASHL WR[0]
 MOV WR[0] TO LS[T5.SUB],
 RETURN

; MAP ADDRESS CORRECTLY
 ; GET DATA FROM LS 6
 ; CLEAR LOW BYTE OF DATA SPECIFIED.
 ; LOW BYTE WILL BE PUT IN BITS 24-31
 ; LATER
 ; GO TO SUBROUTINE TO SHIFT WR 0 EIGHT
 ; PLACES RIGHT
 ; SWAP ORIGINAL DATA SPECIFIED WITH MOD-
 ; IFIED DATA. NOW LS 6 CONTAINS BITS
 ; 8-22 IN BITS 0-14 AS REQUIRED. WR 0
 ; CONTAINS ORIGINAL DATA SPECIFIED
 ; CLEAR ALL BUT LOW BYTE OF ORIGINAL
 ; DATA SPECIFIED
 ; GO TO SUBROUTINE TO SHIFT WR 0 EIGHT
 ; PLACES RIGHT. ON RETURN, WR 0 WILL
 ; HAVE BITS 0-7 IN BITS 24-31, OTHER
 ; BITS ARE CLEAR
 ; NOW LS 6 CONTAINS DATA TO WRITE IN
 ; TB IN CORRECT FORMAT I.E. PA BITS
 ; IN BITS 00-14 AND BYTE OFFSET, MODIFY
 ; PROT A THROUGH PROT D, AND VALID IN
 ; BITS 25-31.

; GET ADDRESS SPECIFIED IN WR 0
 ; SHIFT ADDRESS 2 PLACES LEFT
 ; SHIFT ADDRESS 2 PLACES LEFT
 ; SHIFT ADDRESS 2 PLACES LEFT
 ; SHIFT ADDRESS 2 PLACES LEFT
 ; SHIFT ADDRESS 2 PLACES LEFT
 ; SHIFT LEFT ONE MORE PLACE
 ; NOW ADDRESS IS IN BITS 9-17 OF WR 0
 ; NOW LS 5 CONTAINS CORRECT ADDRESS.
 ; RETURN TO MAIN

```

U 0654, 0864,DC :6142 JSR [SHIFT.UB.ADR] ;SHIFT ADDRESS SPECIFIED FROM LS 5 INTO
:6143 ; CORRECT POSITION AND REPLACE IN LS 5
U 0655, 1C0B,75 :6144 MEM.REQ[READ.UBS.MAP] ADRS[TS.SUB] DT[LONG] ;ISSUE MEMORY REQUEST TO
:6145 ; READ THE UBS MAP AT ADDRESS CONTAINED
:6146 ; IN LS 5
U 0656, 3022,15 :6147 MOV MEM.DATA TO WR[0] ;PLACE RESULT IN WR 0
:6148
U 0657, 452A,15 :6149 BIC LS[0] TO WR[0] ;CLEAR UPPER BYTE OF RESULT (UPPER BYTE

```

U U U U

U U U U U

22 22 2 22 2 2 2 2 2 2

[illegible]


```

ZZ-ENKCF-1.4 : ENKCF.MIC EXAMINE MAIN MEMORY 15-MAR-83 C 12
: ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module
: ENKCF.MIC EXAMINE MAIN MEMORY ROUTINE
Sequence 145
Rev 01.40 Page 139

:6216 .PAGE " EXAMINE MAIN MEMORY ROUTINE"
:6217 .+++
:6218
:6219 .FUNCTIONAL DESCRIPTION:
:6220
:6221 This routine is used to read a longword from main memory at the Phy-
:6222 sical address specified. The address need not be on a longword bound-
:6223 ary. This routine is used by the console on an EXAMINE.MM command.
:6224 The console will have loaded LS 5 with the physical address specified
:6225 before executing this routine.
:6226 The routine will first write CSR 1 to set the INH REP CRD (inhibit
:6227 report correctable read data) bit to prevent an error sum from occurring
:6228 on a single bit error which has been corrected. Then the routine is-
:6229 sues a memory request with memory function=READ.P and DT=longword. It
:6230 then reads the memory location and puts the result in LS 6. It also
:6231 checks for an ERROR SUM and issues a CPU ACK if an error occurred (sin-
:6232 gle bit errors that have been corrected will not cause an ERR SUM). If
:6233 an error occurred, CPU ACK will be set before issueing CPU ATTN to in-
:6234 form the console that an error occurred.
:6235
:6236 .CALLING SEQUENCE:
:6237
:6238 The console loads the address of a pointer to this routine (a JMP in-
:6239 struction at WCS location 55) into the UPC and starts the CPU.
:6240
:6241 .INPUT PARAMETERS:
:6242
:6243 LS 5 - Physical address in main memory to read.
:6244
:6245 .IMPLICIT INPUTS:
:6246
:6247 LS[INH.CRD] - Data to write into CSR 1 to inhibit error sum on a CRD.
:6248 LS[CSR1] - Address for writing CSR 1.
:6249
:6250 .OUTPUT PARAMATERS:
:6251
:6252 LS 6 - Longword data from physical memory address specified.
:6253
:6254 .IMPLICIT OUTPUTS:
:6255
:6256 CPU ACK if an error sum occurs.
:6257
:6258 .SIDE EFFECTS:
:6259
:6260 None
:6261
:6262 .---
:6263
:6264 .EXAMINE.MM:
U 0662, 1D45,F5 :6265 MEM.REQ[WRITE.CSR] ADRS[CSR1] DT[LONG] ;ISSUE MEMORY REQUEST TO
:6266 ; WRITE CSR 1
U 0663, B278,15 :6267 WRITE.MEM LS[INH.CRD] ;SET INH REP CRD IN CSR 1 AND CLEAR
:6268 ; ECC DISABLE
:6269
U 0664, 180B,F5 :6270 MEM.REQ[READ.P] ADRS[TS.SUB] DT[LONG] ;ISSUE MEMORY REQUEST TO

```

	:6271		: READ LONGWORD DATA FROM MAIN MEMORY
	:6272		: PHYSICAL ADDRESS IN LS 5
	:6273		: READ DATA AND PUT IN LS 6
U 0665, 380C,1D	:6274	MOV MEM.DATA TO LS[T6.SUB],	: SKIP NEXT INSTRUCTION IF NO MEMORY
	:6275	SKIP.IF[MEM.REF.OK]	: ERROR (NO ERROR SUM)
U 0666, 10D0,15	:6276	MISC [SET.CP.ACK]	: SET CPU ACK IF AN ERROR OCCURRED
U 0667, 8868,74	:6277	JMP [WAIT.SUB]	: JUMP TO SET ATTN AND WAIT

ZZ-ENKCF-1.4
: ENKCF.MCR
: ENKCF.MIC

ENKCF.MIC

EXAMINE IDC ROUTINE

E 12

15-MAR-83

Fiche 1 Frame E12

Sequence 147

Rev 01.40

Page 141

MICRO2 1M(01)

15-MAR-83

11:46:22

ENKCF Micro-diagnostic for IDC module

EXAMINE IDC ROUTINES

:6278 .page " EXAMINE IDC ROUTINES"

:6279 :+++

:6280

:6281

:6282

:6283

:6284

:6285

:6286

:6287

:6288

:6289

:6290

:6291

:6292

:6293

:6294

:6295

:6296

:6297

:6298

:6299

:6300

:6301

:6302

:6303

:6304

:6305

:6306

:6307

:6308

:6309

:6310

:6311

:6312

:6313

:6314

:6315

EXAMINE.ICSR:

MISC [SET.PORT.I.0]

PORT.READ PORT.ADRS[CSR]

JMP [READ.IDC]

;SELECT PORT TO BUS

;SEND THE READ COMMAND

;READ THE DATA

EXAMINE.IDAR:

MISC [SET.PORT.I.0]

PORT.READ PORT.ADRS[DAR]

JMP [READ.IDC]

;SELECT PORT TO BUS

;SEND THE READ COMMAND

;READ THE DATA

EXAMINE.DBUF:

MISC [SET.PORT.I.0]

PORT.READ PORT.ADRS[DATA.WORD]

JMP [READ.IDC]

;SELECT PORT TO BUS

;SEND THE READ COMMAND

;READ THE DATA

EXAMINE.PATT:

MISC [SET.PORT.I.0]

PORT.READ PORT.ADRS[PATTERN]

;SELECT PORT TO BUS

;SEND THE READ COMMAND

U 0668, 9090,15

U 0669, 9780,15

U 066A, 0867,64

U 066B, 9090,15

U 066C, 1784,15

U 066D, 0867,64

U 066E, 9090,15

U 066F, 9780,15

U 0670, 0867,64

U 0671, 9090,15

U 0672, 1790,15

U 0673, 0867,64	:6333	JMP [READ.IDC]	;READ THE DATA
	:6334		
	:6335	EXAMINE.POSIT:	
U 0674, 9090,15	:6336	MISC [SET.PORT.I.0]	;SELECT PORT TO BUS
U 0675, 9794,15	:6337	PORT.READ PORT.ADRS[POSITION]	;SEND THE READ COMMAND
	:6338		
	:6339		
	:6340	READ.IDC:	
U 0676, 3A0C,15	:6341	MOV PORT TO LS[T6.SUB]	;READ DATA AND PUT IN LS 6
U 0677, 1080,15	:6342	MISC [SET.ACC.I.0]	;RETURN SELECT TO ACCELERATOR
U 0678, 8868,74	:6343	JMP [WAIT.SUB]	;JUMP TO SET ATTN AND WAIT

:6344 :.page " DEPOSIT IDC ROUTINES"
:6345 :+++

:6346 :
:6347 : FUNCTIONAL DESCRIPTION:

:6348 :
:6349 : The following routines are used to write data to the optional IDC
:6350 : module's registers. The data is taken from LS 6 which is previously
:6351 : loaded by the monitor when the DEPOSIT command is executed.
:6352 :

:6353 : CALLING SEQUENCE:

:6354 :
:6355 : The console loads the address of a pointer to this routine (a JMP in-
:6356 : struction at WCS location 5F through 61) into the UPC and starts the
:6357 : CPU.
:6358 :

:6359 : INPUT PARAMETERS:

:6360 :
:6361 : LS 6 - Data pattern to write.
:6362 :

:6363 : IMPLICIT INPUTS:

:6364 :
:6365 : None
:6366 :

:6367 : OUTPUT PARAMATERS:

:6368 :
:6369 : None
:6370 :

:6371 : IMPLICIT OUTPUTS:

:6372 :
:6373 : None
:6374 :

:6375 : SIDE EFFECTS:

:6376 :
:6377 : None
:6378 :

:6379 :---
:6380 :

:6381 :DEPOSIT.ICSR:

U 0679, 360C,15 :6382 :MOV LS[T6.SUB] TO WR[0] ;GET DATA PATTERN TO WRITE
U 067A, 17A0,15 :6383 :PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE TO IDC
U 067B, 8868,74 :6384 :JMP [WAIT.SUB] ;JUMP TO SET ATTN AND WAIT
:6385 :
:6386 :

:6387 :DEPOSIT.IDAR:

U 067C, 360C,15 :6388 :MOV LS[T6.SUB] TO WR[0] ;GET DATA PATTERN TO WRITE
U 067D, 97A4,15 :6389 :PORT.WRITE PORT.ADRS[DAR] WITH WR[0] ;WRITE TO IDC
U 067E, 8868,74 :6390 :JMP [WAIT.SUB] ;JUMP TO SET ATTN AND WAIT
:6391 :
:6392 :

:6393 :DEPOSIT.DBUF:

U 067F, 360C,15 :6394 :MOV LS[T6.SUB] TO WR[0] ;GET DATA PATTERN TO WRITE
U 0680, 17AC,15 :6395 :PORT.WRITE PORT.ADRS[DATA.WORD] WITH WR[0] ;WRITE TO IDC
U 0681, 8868,74 :6396 :JMP [WAIT.SUB] ;JUMP TO SET ATTN AND WAIT
:6397 :
:6398 :

U 0682, 17C0,15	:6399	CLEAR.FIFO.ADDR:	
U 0683, 8868,74	:6400	PORT.CONTROL [CLEAR.FIFO.CNTR]	;CLEAR ADDRESS IN FIFO A OR FIFO B
	:6401	JMP [WAIT.SUB]	;JUMP TO SET ATTN AND WAIT
	:6402		
	:6403		
U 0684, 17D8,15	:6404	SELECT.FIFO.A:	
U 0685, 8868,74	:6405	PORT.CONTROL [SEL.FIFO.A]	;SELECT FIFO A
	:6406	JMP [WAIT.SUB]	;JUMP TO SET ATTN AND WAIT
	:6407		
	:6408		
U 0686, 97DC,15	:6409	SELECT.FIFO.B:	
	:6410	PORT.CONTROL [SEL.FIFO.B]	;SELECT FIFO B
	:6411		
	:6412		
U 0687, 10E0,15	:6413	WAIT.SUB:	
	:6414	MISC [SET.CP.ATTN]	;ISSUE CPU ATTN TO CONSOLE
U 0688, 8868,84	:6415	WAIT.DE.EX:	
		JMP [WAIT.DE.EX]	;LOOP SELF UNTIL CONSOLE TAKES CONTROL

U U U U U U U U


```

:6467 .PAGE  " RESTORE WR ROUTINE"
:6468 :+++
:6469 :
:6470 : FUNCTIONAL DESCRIPTION:
:6471 :
:6472 :     This routine restores the 2901 working registers save in LS 1-4. The
:6473 :     console calls this routine before the CPU is restarted after it has
:6474 :     been halted by the console.
:6475 :     The routine simply moves the data from LS locations 1-4 to WRs 0-3
:6476 :     and then issues a CPU ATTN to the console. A JMP self is executed next
:6477 :     waiting for the console to take control.
:6478 :
:6479 : CALLING SEQUENCE:
:6480 :
:6481 :     The console loads the address of a pointer to this routine (a JMP in-
:6482 :     struction at WCS location 47) into the UPC and starts the CPU.
:6483 :
:6484 : INPUT PARAMETERS:
:6485 :
:6486 :     LS 1 contains the data to be restored in WR 0.
:6487 :     LS 2 contains the data to be restored in WR 1.
:6488 :     LS 3 contains the data to be restored in WR 2.
:6489 :     LS 4 contains the data to be restored in WR 3.
:6490 :
:6491 : IMPLICIT INPUTS:
:6492 :
:6493 :     None
:6494 :
:6495 : OUTPUT PARAMATERS:
:6496 :
:6497 :     WR 0 gets data from LS 1.
:6498 :     WR 1 gets data from LS 2.
:6499 :     WR 2 gets data from LS 3.
:6500 :     WR 3 gets data from LS 4.
:6501 :
:6502 : IMPLICIT OUTPUTS:
:6503 :
:6504 :     None
:6505 :
:6506 : SIDE EFFECTS:
:6507 :
:6508 :     None
:6509 :
:6510 : ---
:6511 :
:6512 : RESTORE.WR:
U 068E, B602,15 :6513     MOV LS[SAV.WR0] TO WR[0]           :RESTORE WR 0
U 068F, 3604,95 :6514     MOV LS[SAV.WR1] TO WR[1]           :RESTORE WR 1
U 0690, B607,15 :6515     MOV LS[SAV.WR2] TO WR[2]           :RESTORE WR 2
U 0691, B609,95 :6516     MOV LS[SAV.WR3] TO WR[3]           :RESTORE WR 3
U 0692, 8868,74 :6517     JMP [WAIT.SUB]                   :JUMP TO SET ATTN AND WAIT

```

```
:6518 .PAGE 'WCS SUBROUTINES FOR CPU USE'
:6519 .TOC ' ERROR ROUTINE'
:6520 .+++
:6521
:6522 .FUNCTIONAL DESCRIPTION:
:6523
:6524 .This routine is used to detect and report diagnostic errors occurring
:6525 .during WCS based micro-diagnostics. The WCS micro-code sets up the
:6526 .parameters listed below and calls this routine. The routine compares
:6527 .WR (expected data) and WR 0 (received data) according to the error
:6528 .mask. Bits set in the mask indicate bits that are not checked for
:6529 .errors.
:6530 .If WR 0 and WR 1 are equal (no error) the routine executes a return+1
:6531 .to the calling point. The only exception to this is if loop on error
:6532 .is set. In that case, the error number is checked to see if that error
:6533 .occurred previously. If so, then the error loop is forced by doing a
:6534 .return. This will lock an error loop in on intermittent errors.
:6535 .If WR 0 and WR 1 are not equal (an error) then bit 8 is set in the
:6536 .CONTROL AND STATUS word, expected and received data is set up in LS for
:6537 .printout, and flags are checked in the ERROR CONTROL word. This con-
:6538 .trols whether printouts are disabled, loop on error is enabled, etc.
:6539 .After printing the error, a return+1 is made to the calling point.
:6540 .Loop on error causes a return rather than return+1 to cause an error
:6541 .loop. Also CPU ACKNOWLEDGE is toggled at the end of the routine for
:6542 .a scope sync point.
:6543
:6544 .CALLING SEQUENCE:
:6545
:6546 .JSR [CHECK.RESULT]
:6547
:6548 .INPUT PARAMETERS:
:6549
:6550 .WR[0] = Received data i.e. the actual result of the test
:6551 .WR[1] = Expected data i.e. what the result should have been
:6552
:6553 .IMPLICIT INPUTS:
:6554
:6555 .LS[ERROR.MASK] = Used to mask out bits that are not to be tested
:6556 .LS[ERROR.NUMBER] = Current error number
:6557 .LS[PREVIOUS.ERROR] = Error number of previous data
:6558 .LS[CONTROL.STATUS] = Control and Status word. Contains information
:6559 .written by the CPU to inform the monitor what to do.
:6560 .LS[ERROR.CONTROL] = Information such as loop-on-error, bell-on-error,
:6561 .no error report, and halt on error.
:6562
:6563 .OUTPUT PARAMATERS:
:6564
:6565 .NONE
:6566
:6567 .IMPLICIT OUTPUTS:
:6568
:6569 .LS[CONTROL.STATUS] = Control and Status with new status information
:6570 .LS[PREVIOUS.ERROR] = Last error number (modified only if error and
:6571 .loop on error is enabled)
:6572 .LS[EXPECTED.DATA] = Expected result if an error was detected
```

```

:6573      :      LS[RECEIVED.DATA] = Actual result if an error was detected
:6574
:6575      :      SIDE EFFECTS:
:6576
:6577      :      WR[0], WR[1], and the Q reg are modified and are not restored before
:6578      :      returning.
:6579      :      If loop-on-error is enabled and an error loop is to be executed, CPU
:6580      :      ACKNOWLEDGE will be toggled for a scope sync.
:6581
:6582      :      ---
:6583
:6584      :      CHECK.RESULT:
U 0693, 458A,15 :6585      BIC LS[ERROR.MASK] TO WR[0]      ;MASK OFF NOT TESTED BITS (CLEAR THEM)
:6586      :      ; FOR RECEIVED DATA
U 0694, C58A,95 :6587      BIC LS[ERROR.MASK] TO WR[1]      ;DITTO FOR EXPECTED DATA
:6588
:6589      :      XOR WR[0] WITH WR[1] TO Q,      ;CMP RECEIVED DATA IN WR[0] WITH
U 0695, AB80,B5 :6590      DT(LONG)&SET.ALU.CC      ; EXPECTED DATA IN WR[1] FOR ERROR
U 0696, 8869,F1 :6591      JMP.IF[NEQ] TO [ERROR]      ;JUMP IF ERROR
:6592
U 0697, 3690,15 :6593      MOV LS[ERROR.CONTROL] TO WR[0]    ;GET ERROR CONTROL WORD FROM LS
:6594      :      BIT WR[0] WITH LS[LOE],      ;SEE IF LOOP ON ERROR IS ENABLED
U 0698, 5940,35 :6595      DT(LONG)&SET.ALU.CC      ; SET CONDITION CODES
U 0699, DB00,01 :6596      SKIP.IF[BIT.SET]      ;SKIP IF IT IS
U 069A, DB00,16 :6597      RETURN+1      ;ELSE RETURN+1 (NO ERROR AND NO LOOP)
:6598
U 069B, 36A0,15 :6599      MOV LS[PREVIOUS.ERROR] TO WR[0]    ;IF NO ERROR BUT LOOP ON ERROR IS
:6600      :      ; ENABLED, SEE IF THERE WAS A
:6601      :      ; PREVIOUS ERROR
:6602
U 069C, CD82,35 :6603      XOR WR[0] WITH LS[ERROR.NUMBER]    ;TO Q,
:6604      :      DT(LONG)&SET.ALU.CC      ;IS THIS THE SAME SPOT THAT HAD AN
U 069D, 086B,91 :6605      JMP.IF[NEQ] TO [RET+1]      ; ERROR BEFORE? (INTERMITTANT ERROR)
:6606      :      ;JUMP IF NOT TO RETURN WITHOUT ERROR
U 069E, 086B,C4 :6607      JMP [RET]      ; LOOPING (RETURN+1)
:6608      :      ;ELSE CONTINUE TO LOOP (WE WILL LOOP
:6609      :      ; ON ERROR EVEN IF IT GOES AWAY)
:6610
:6611      :      ERROR:
U 069F, 3E86,15 :6611      MOV WR[0] TO LS[RECEIVED.DATA]    ;PUT RESULT OF TEST IN LS FOR PRINTOUT
U 06A0, 3690,15 :6612      MOV LS[ERROR.CONTROL] TO WR[0]    ;GET ERROR CONROL BITS TO CHECK FOR
:6613      :      ; LOOP COMMAND
:6614      :      BIT WR[0] WITH LS[LOOP.COMMAND], ;SEE IF BIT 6 SET
U 06A1, 594C,35 :6615      DT(LONG)&SET.ALU.CC      ; SET CONDITION CODES
U 06A2, 086B,C1 :6616      JMP.IF[BIT.SET] TO [RET]      ;GO LOOP IF SET (FAST LOOP)
:6617
U 06A3, 3E84,95 :6618      MOV WR[1] TO LS[EXPECTED.DATA]    ;PUT EXPECTED DATA IN LS FOR PRINTOUT
:6619
U 06A4, 3680,95 :6620      MOV LS[CONTROL.STATUS] TO WR[1]    ;GET CONTROL AND STATUS WORD FROM LS
U 06A5, C532,95 :6621      BIC LS[#FE7FFFFFFF] TO WR[1]      ;CLEAR ALL BUT BITS 23&24 IN CONTROL
:6622      :      ; AND STATUS WORD
U 06A6, C750,95 :6623      BIS LS[ERROR] TO WR[1]      ;SET BIT 8 IN CONTROL AND STATUS WORD
:6624      :      ; (INDICATES AN ERROR WAS DETECTED)
U 06A7, BE80,95 :6625      MOV WR[1] TO LS[CONTROL.STATUS]    ;WRITE UPDATED CONTROL AND STATUS WORD
:6626      :      ; BACK TO LS[40]
:6627

```

ZZ-ENKCF-1.4	:	ENKCF.MIC	ERROR ROUTINE	M 12	15-MAR-83	Fiche 1 Frame M12	Sequence 155	
:	:	ENKCF.MCR	MICRO2 1M(01)	15-MAR-83	11:46:22	ENKCF Micro-diagnostic for IDC module	Rev 01.40	Page 149
:	:	ENKCF.MIC	ERROR ROUTINE					

U 06A8, D944,35	:6628	BIT WR[0] WITH LS[BELL],	;SEE IF BELL ON ERROR IS SET (WR 0 STILL
U 06A9, 886A,D9	:6629		; CONTAINS ERROR CONTROL WORD)
U 06AA, 10E0,15	:6630	DT(LONG)&SET.ALU.CC	; SET CONDITION CODES
	:6631	JMP.IF[BITS.CLR] TO [CHECK.NER]	;IF BELL ON ERROR IS NOT SET, JUMP TO
	:6632		; SEE IF ERROR REPORTING IS INHIBITED
	:6633	MISC [SET.CP.ATTN]	;ELSE SET CPU ATTN (RING MY BELL)
	:6634		
U 06AB, 086A,B4	:6635	WAIT.1: JMP [WAIT.1]	;WAIT FOR CONSOLE TO STOP ME
U 06AC, 086B,64	:6636	JMP [LOOP]	;GO TO CHECK LOOP-ON-ERROR AFTER
	:6637		; CONSOLE HAS FINISHED
	:6638		
	:6639	CHECK.NER:	
	:6640	BIT WR[0] WITH LS[NER],	;IF NO BELL SEE IF ERROR REPORT IS
	:6641		; INHIBITED
U 06AD, D942,35	:6642	DT(LONG)&SET.ALU.CC	; SET CONDITION CODES
U 06AE, 886B,21	:6643	JMP.IF[BIT.SET] TO [CHECK.HALT]	;JUMP IF ERROR REPORT IS INHIBITED TO
	:6644		; CHECK FOR HALT ON ERROR
	:6645		
U 06AF, 10E0,15	:6646	MISC [SET.CP.ATTN]	;SET CPU ATTN (REPORT ERROR)
U 06B0, 086B,04	:6647	WAIT.2: JMP [WAIT.2]	;WAIT FOR CONSOLE TO STOP ME
U 06B1, 086B,64	:6648	JMP [LOOP]	;GO TO CHECK LOOP-ON-ERROR AFTER
	:6649		; CONSOLE HAS FINISHED
	:6650		
	:6651	CHECK.HALT:	
	:6652	BIT WR[0] WITH LS[HOE],	;SEE IF HALT ON ERROR IS ENABLED
U 06B2, 5946,35	:6653	DT(LONG)&SET.ALU.CC	; SET CONDITION CODES
U 06B3, 886B,69	:6654	JMP.IF[BITS.CLR] TO [LOOP]	;JUMP IF NOT TO CHECK LOOP-ON-ERROR
	:6655		
U 06B4, 10E0,15	:6656	MISC [SET.CP.ATTN]	;ELSE SET CPU ATTN (HALT ON ERROR)
U 06B5, 086B,54	:6657	WAIT.3: JMP [WAIT.3]	;WAIT FOR CONSOLE TO STOP ME
	:6658		
U 06B6, 3690,15	:6659	LOOP: MOV LS[ERROR.CONTROL] TO WR[0]	;GET ERROR CONTROL BITS (NER, HOE ETC.)
	:6660	BIT WR[0] WITH LS[LOE],	;SEE IF LOOP-ON-ERROR
U 06B7, 5940,35	:6661	DT(LONG)&SET.ALU.CC	; SET CONDITION CODES
U 06B8, DB00,01	:6662	SKIP.IF[BIT.SET]	; SKIP IF LOOP ON ERROR IS ENABLED
	:6663		
U 06B9, DB00,16	:6664	RET+1: RETURN+1	;ELSE RETURN+1 FOR NO ERROR LOOP
	:6665		
U 06BA, 3682,15	:6666	MOV LS[ERROR.NUMBER] TO WR[0]	;GET ERROR NUMBER IF LOOPING
	:6667		
U 06BB, BEA0,15	:6668	MOV WR[0] TO LS[PREVIOUS.ERROR]	;AND SAVE IT IN PREVIOUS ERROR
	:6669		
U 06BC, 10D0,15	:6670	RET: MISC [SET.CP.ACK]	;SET CPU ACKNOWLEDGE FOR SCOPE SYNC
	:6671	MISC [CLR.CP.ATTN.AND.ACK],	;AND THEN CLEAR IT
U 06BD, 10C0,14	:6672	RETURN	; RETURN TO TEST AND LOOP ON ERROR
	:6673		

ZZ
:
:
U
U
U

```

:6674 .PAGE
:6675
:6676
:6677
:6678 ERR.NO.TEST:
U 06BE, DB00,15 :6679 NOP ;NOP WILL CAUSE A SEQUENCE ERROR
U 06BF, B65E,15 :6680 MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT 15 IN WR[0] FOR CONTROL AND
:6681 ; STATUS WORD
U 06C0, 3E80,15 :6682 MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
:6683 ; BIT 15 INDICATES START OF TEST TO
:6684 ; CONSOLE
U 06C1, 8868,74 :6685 JMP [WAIT.SUB] ;JUMP TO SET ATTN AND WAIT
:6686 .TOC "START TRANSFER ROUTINE"
:6687 :+++
:6688 This routine loads the LS with information necessary for these tests
:6689 to run. It will then issue a CPU ATTN with the control and status word
:6690 clear to indicate that all transfers are complete.
:6691
:6692 RESULT: LS LOADED WITH CONSTANTS. MONITOR INFORMED TRANSFERS OVER.
:6693 :---
:6694
:6695 TRANSFER.POINT:
:6696
:6697
U 06C2, 2F80,15 :6698 CLR WR[0] ;START WITH 0 IN WR[0]
U 06C3, BFFE,15 :6699 MOV WR[0] TO LS[PSL.HW] ;MAKE SURE COMPAT MODE IS CLEAR
:6700
U 06C4, 2040,15 :6701 INC WR[0] ;NOW HAVE 1 IN WR[0]
U 06C5, 23D0,15 :6702 ASHL WR[0] ;10(B)
U 06C6, 2040,15 :6703 INC WR[0] ;11(B)
U 06C7, 23D0,15 :6704 ASHL WR[0] ;110(B)
U 06C8, 2040,15 :6705 INC WR[0] ;111(B)
U 06C9, 23D0,15 :6706 ASHL WR[0] ;1110(B)
U 06CA, 23D0,15 :6707 ASHL WR[0] ;1 1100(B)
U 06CB, 23D0,15 :6708 ASHL WR[0] ;11 1000(B)
U 06CC, 23D0,15 :6709 ASHL WR[0] ;111 0000(B) NOW HAVE 70(H) IN WR 0.
:6710 ; THIS IS CORRECT START ADDRESS OF DATA
:6711 ; SECTION (INCLUDING LONGWORD COUNT,
:6712 ; DESTINATION, AND DESTINATION ADDRESS)
U 06CD, 3E0E,15 :6713 MOV WR[0] TO LS[TRANSFER.POINTER] ;LOAD LS 7 WITH POINTER TO TRANSFER
:6714 ; DATA SECTION.
:6715
U 06CE, 2F80,15 :6716 CLR WR[0] ;0 IN WR 0
U 06CF, 2040,15 :6717 INC WR[0] ;1 IN WR 0
U 06D0, 23D0,15 :6718 ASHL WR[0] ;10(B)
U 06D1, 23D0,15 :6719 ASHL WR[0] ;100(B)
U 06D2, 23D0,15 :6720 ASHL WR[0] ;1000(B)
U 06D3, 23D0,15 :6721 ASHL WR[0] ;1 0000(B)
U 06D4, 23D0,15 :6722 ASHL WR[0] ;10 0000(B)
U 06D5, 23D0,15 :6723 ASHL WR[0] ;100 0000(B)
U 06D6, 23D0,15 :6724 ASHL WR[0] ;1000 0000(B)
U 06D7, 23D0,15 :6725 ASHL WR[0] ;1 0000 0000(B)
U 06D8, 23D0,15 :6726 ASHL WR[0] ;10 0000 0000(B)
U 06D9, 23D0,15 :6727 ASHL WR[0] ;100 0000 0000(B)
U 06DA, 23D0,15 :6728 ASHL WR[0] ;1000 0000 0000(B)

```

```

B 13
22-ENKCF-1.4 : ENKCF.MIC START TRANSFER ROUTINE 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module
: ENKCF.MCR
: ENKCF.MIC
Sequence 157 Page 151
:
:
:

U 06DB, 23D0,15 :6729 ASHL WR[0] ;1 0000 0000 0000(B)
U 06DC, 23D0,15 :6730 ASHL WR[0] ;10 0000 0000 0000(B)
U 06DD, 3E80,15 :6731 MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 13 IN LS CONTROL AND STATUS
:6732 ; WORD (BIT 13 INDICATES CONSOLE MUST
:6733 ; PERFORM A DATA TRANSFER)
U 06DE, 10E0,15 :6734 MISC [SET.CP.ATTN] ;ISSUE CPU ATTN TO CONSOLE (DO FIRST LS
:6735 ; TRANSFER TO LOCATIONS 0 THROUGH 77(H))
:6736 WAIT.XFER1:
U 06DF, 086D,F4 :6737 JMP [WAIT.XFER1] ;LOOP HERE WAITING FOR CONSOLE TO RE-
:6738 ; SPOND
:6739
U 06E0, 36CA,15 :6740 MOV LS[#162(H)] TO WR[0] ;GET START ADDRESS OF SECOND HALF OF LS
:6741 ; TRANSFER
U 06E1, 3E0E,15 :6742 MOV WR[0] TO LS[TRANSFER.POINTER] ;LOAD START ADDRESS IN LS FOR CONSOLE
U 06E2, 365A,15 :6743 MOV LS[XFER.DATA] TO WR[0] ;BIT 13 INDICATES TO DO DATA TRANSFER
U 06E3, 3E80,15 :6744 MOV WR[0] TO LS[CONTROL.STATUS] ;SET UP CONTROL AND STATUS WORD
U 06E4, 10E0,15 :6745 MISC [SET.CP.ATTN] ;ISSUE CPU ATTN TO CONSOLE (DO SECOND LS
:6746 ; TRANSFER TO LOCATIONS 80 THROUGH F7(H))
:6747 WAIT.XFER2:
U 06E5, 086E,54 :6748 JMP [WAIT.XFER2] ;LOOP HERE WAITING FOR CONSOLE TO RE-
:6749 ; SPOND
:6750
U 06E6, 9B9D,75 :6751 MEM.REQ[READ.V.RCHK.IFILL] ADRS[ZERO] DT[LONG] ;THIS INSTRUCTION USED TO FREE MEMORY
:6752 ; IF HUNG WAITING FOR A DATA RCVD.
:6753
U 06E7, 3022,15 :6754 MOV MEM.DATA TO WR[0] ;THIS INSTRUCTION USED TO FREE MEMORY
:6755 ; IF HUNG IN A OCTA FLOW
U 06E8, 3022,15 :6756 MOV MEM.DATA TO WR[0] ;NEEDS 4 OF THESE
U 06E9, 3022,15 :6757 MOV MEM.DATA TO WR[0]
U 06EA, 3022,15 :6758 MOV MEM.DATA TO WR[0] ;DONE WITH MEMORY INITIALIZATION
:6759
U 06EB, 17CC,15 :6760 PORT.CONTROL [CLEAR.IDC] ;INITIALIZE IDC TO IDLE LOOP
U 06EC, 17CC,15 :6761 PORT.CONTROL [CLEAR.IDC]
U 06ED, 17CC,15 :6762 PORT.CONTROL [CLEAR.IDC]
:6763
U 06EE, 17D4,15 :6764 PORT.CONTROL [CLEAR.AUTO] ;CLEAR AUTOMODE
:6765
U 06EF, 3678,15 :6766 MOV LS[BIT28] TO WR[0]
U 06F0, B64E,95 :6767 MOV LS[BIT7] TO WR[1]
U 06F1, AEC2,15 :6768 BIS WR[1] TO WR[0]
U 06F2, 17A0,15 :6769 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;SET WRITE INHIBIT IN IDC CSR
:6770 ; AND SET CRDY
:6771
U 06F3, 3660,15 :6772 MOV LS[INTERUPT.EN] TO WR[0] ;SET BIT 16 IN WR
U 06F4, 3E80,15 :6773 MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 16 TO CLEAR ANY INTERRUPTS
:6774 ; AND INFORM CONSOLE THAT XFERS ARE
:6775 ; DONE
U 06F5, 10E0,15 :6776 MISC [SET.CP.ATTN] ;CALL CONSOLE
:6777 WAIT.XFER:
U 06F6, 886F,64 :6778 JMP [WAIT.XFER] ;WAIT FOR CONSOLE TO RESPOND
U 06F7, 5B00,14 :6779 RETURN ;USED WHEN TRANSFER IS CALLED FROM
:6780 ; MICRO-DIAGNOSTIC ONLY
:6781

```

22

:

:

U


```

:6785 .PAGE  "SUBROUTINES FOR IDC MICRODIAGNOSTIC"
:6786 .TOC  "DISK DRIVE SIZING ROUTINE"
:6787 :+++
:6788 :
:6789 :FUNCTIONAL DESCRIPTION:
:6790 :
:6791 :    This subroutine will check each drive <3:0> and determine by looking
:6792 :    at bit <26> of the CSR whether the drive attached is an RL02 or an R80.
:6793 :    If no disk drive is attached, then CSR bit <26>, RL02 is asserted.
:6794 :    When an R80 is found, no further sizing is done because it is assumed
:6795 :    that the other drives are RL02s. When the sizing has been completed,
:6796 :    a message will be printed indicating what was found by this subroutine.
:6797 :    If this information is incorrect, then THE DISK SIZING TEST should
:6798 :    be run to find the error.
:6799 :
:6800 :CALLING SEQUENCE:
:6801 :
:6802 :    JSR [DRIVE.SIZE]
:6803 :
:6804 :INPUT PARAMETERS:
:6805 :
:6806 :    NONE
:6807 :
:6808 :IMPLICIT INPUTS:
:6809 :
:6810 :    NONE
:6811 :
:6812 :OUTPUT PARAMATERS:
:6813 :
:6814 :    NONE
:6815 :
:6816 :IMPLICIT OUTPUTS:
:6817 :
:6818 :    The drive information will be stored in LS[DRIVE.STATUS] BITS <3:0>.
:6819 :    A one in any bit position indicates that an R80 was found on that
:6820 :    drive number. Example: LS[DRIVE.STATUS] = 00000001 indicates that an
:6821 :    R80 was found on DRIVE 0.
:6822 :
:6823 :SIDE EFFECTS:
:6824 :
:6825 :    WR[0] and WR[1] will be modified
:6826 :    LS[T6.SUB] and LS[T7.SUB] will be modified
:6827 :    LS[CONTROL.STATUS] will be modified
:6828 :---
:6829 :*****
:6830 :DRIVE.SIZE:
:6831 :
:6832 :    CLR LS[T6.SUB] ;DRV NUM FOR PRINTOUT
:6833 :    CLR LS[T7.SUB] ;USED IF R80 FOUND FOR PRINTOUT
:6834 :
:6835 :TEST.DRIVE:
:6836 :    MOV LS[T6.SUB] TO WR[0] ;GET THE DRIVE NUMBER
:6837 :    MOV LS[#8] TO WR[1] ;SHIFT THE DRIVE NUMBER INTO
:6838 :
:6839 :SHIFT:
:        ASHL WR[0] ; BITS <9:8> FOR THE WRITE TO
    
```

```

U 0700, 650C,15
U 0701, E50E,15
U 0702, 360C,15
U 0703, 3646,95
U 0704, 23D0,15
    
```


E 13
Fiche 1 Frame E13
Sequence 160
Page 154

ZZ-ENKCF-1.4 : ENKCF.MIC DISK DRIVE SIZING R15-MAR-83
: ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40
: ENKCF.MIC DISK DRIVE SIZING ROUTINE

```

:6840          DEC WR[1],          ; THE CSR
U 0705, A102,B5 :6841          DT(LONG)&SET.ALU.CC
U 0706, 8870,41 :6842          JMP.IF[NEQ] TO [SHIFT]
:6843          ;DRIVE NUM IS NOW IN BITS<9:8>
:6844
U 0707, 4778,15 :6845          BIS LS[BIT28] TO WR[0]          ;SET THE TIMEOUT INHIBIT BIT
U 0708, 17A0,15 :6846          PORT.WRITE PORT.ADRS[CSR] WITH WR[0]      ;WRITE THE CSR
:6847
U 0709, 9090,15 :6848          MISC [SET.PORT.I.O]          ;SELECT THE IDC
U 070A, 9780,15 :6849          PORT.READ PORT.ADRS[CSR]          ;SET UP TO READ THE CSR
U 070B, 3B32,15 :6850          MOV PORT TO LS[PORT0]          ;READ THE CSR,SAVE IN LS
U 070C, 3732,15 :6851          MOV LS[PORT0] TO WR[0]
:6852
:6853          BIT LS[BIT26] WITH WR[0],          ;WAS THE DRIVE AN R80?
U 070D, D974,35 :6854          DT(LONG)&SET.ALU.CC
U 070E, 8871,A1 :6855          JMP.IF[BIT.SET] TO [R80.FOUND]      ;JUMP IF DRIVE IS AN R80
:6856
U 070F, B788,15 :6857          MOV LS[#3] TO WR[0]          ;LAST DRIVE BEEN TESTED?
:6858          CMP WR[0] WITH LS[T6.SUB],
U 0710, 4F0C,35 :6859          DT(LONG)&SET.ALU.CC
U 0711, 8871,49 :6860          JMP.IF[EQL] TO [DONE]          ;JMP IF ALL DRVS BEEN TESTED
:6861
U 0712, FF0C,15 :6862          INC LS[T6.SUB]          ;INCREMENT THE DRIVE NUMBER
U 0713, 8870,24 :6863          JMP [TEST.DRIVE]          ;GO TEST NEXT DRIVE NUMBER
:6864
:6865          DONE:
:6866
U 0714, 364E,15 :6867          MOV LS[PRINT.R80] TO WR[0]          ;SET UP TO PRINT NO R80s
U 0715, 3E80,15 :6868          MOV WR[0] TO LS[CONTROL.STATUS]
U 0716, 10E0,15 :6869          MISC [SET.CP.ATTN]          ;ISSUE CPU ATTN TO CONSOLE
:6870          WAIT.DRIVE.SIZE.1:
U 0717, 0871,74 :6871          JMP [WAIT.DRIVE.SIZE.1]          ;LOOP HERE WAITING FOR
:6872
U 0718, 2F82,95 :6873          CLR WR[1]          ;SAVE NO R80 FOUND
:6874
U 0719, 0872,B4 :6875          JMP [SAVE.INFO]          ;SAVE THE INFO THAT WAS FOUND
:6876
:6877
:6878
:6879          R80.FOUND:
U 071A, B640,15 :6880          MOV LS[#1] TO WR[0]
U 071B, 3E0E,15 :6881          MOV WR[0] TO LS[T7.SUB]          ;INDICATES AN R80 FOUND TO THE
:6882          ; PRINT ROUTINE
:6883          ; DRV NUM IS IN LS[T6.SUB]
U 071C, 364E,15 :6884          MOV LS[PRINT.R80] TO WR[0]          ;SET UP TO PRINT THAT AN
U 071D, 3E80,15 :6885          MOV WR[0] TO LS[CONTROL.STATUS]      ; R80 WAS FOUND
:6886
U 071E, 10E0,15 :6887          MISC [SET.CP.ATTN]          ;ISSUE CPU ATTN TO CONSOLE
:6888          WAIT.DRIVE.SIZE.2:
U 071F, 8871,F4 :6889          JMP [WAIT.DRIVE.SIZE.2]          ; TO GO PRINT MESSAGE
:6890          ;LOOP HERE WAITING FOR
:6891          ; CONSOLE TO RESPOND
U 0720, 3640,95 :6892          MOV LS[BIT0] TO WR[1]          ;SAVE DRIVE 0 IN WR[1]
U 0721, 360C,15 :6893          MOV LS[T6.SUB] TO WR[0]          ;GET THE DRIVE NUM
:6894          CMP LS[ZERO] WITH WR[0],          ;DRV NUM = 0?

```

F 13

ZZ-ENKCF-1.4 : ENKCF.MIC DISK DRIVE SIZING R15-MAR-83 Fiche 1 Frame F13 Sequence 161
 : ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40 Page 155
 : ENKCF.MIC DISK DRIVE SIZING ROUTINE

```

U 0722, CE9C,35 :6895      DT(LONG)&SET.ALU.CC
U 0723, 8872,B9 :6896      JMP.IF[EQL] TO [SAVE.INFO]           ;JMP IF DRIVE 0
:6897
U 0724, B642,95 :6898      MOV LS[BIT1] TO WR[1]           ;SAVE DRIVE 1 IN WR[1]
:6899      CMP LS[#1] WITH WR[0],       ;DRV NUM = 1?
U 0725, 4E40,35 :6900      DT(LONG)&SET.ALU.CC
U 0726, 8872,B9 :6901      JMP.IF[EQL] TO [SAVE.INFO]           ;JMP IF DRIVE 1
:6902
U 0727, B644,95 :6903      MOV LS[BIT2] TO WR[1]           ;SAVE DRIVE 2 IN WR[1]
:6904      CMP LS[#2] WITH WR[0],       ;DRV NUM = 2?
U 0728, CE42,35 :6905      DT(LONG)&SET.ALU.CC
U 0729, 8872,B9 :6906      JMP.IF[EQL] TO [SAVE.INFO]           ;JMP IF DRIVE 2
:6907
U 072A, 3646,95 :6908      MOV LS[BIT3] TO WR[1]           ;SAVE DRIVE 3 IN WR[1]
:6909
:6910
:6911
U 072B, C74E,95 :6912      SAVE.INFO:
:6913      BIS LS[BIT7] TO WR[1]           ;SET BIT 7 TO INDICATE THAT
:6914      MOV WR[1] TO LS[DRIVE.STATUS] ; THIS TEST HAS BEEN RUN
:6915
U 072D, 5B00,14 :6916      RETURN
:6917
:6918
  
```

ZZ-ENKCF-1.4
: ENKCF.MCR
: ENKCF.MIC

ENKCF.MIC

DISK DRIVE SIZING R15-MAR-83
MICRO2 1M(01) 15-MAR-83 11:46:22
DISK DRIVE SIZING ROUTINE

G 13

Fiche 1 Frame G13

Sequence 162

Rev 01.40 Page 156

```
:6919 .PAGE
:6920 .TOC  " INIT IDC AND GO TO DIAG.IDLE ROUTINE "
:6921 .+++
:6922
:6923 .FUNCTIONAL DESCRIPTION:
:6924
:6925     This routine is used to force the IDC microcode to the Diagnostic
:6926     Idle Loop (DIAG.IDLE) which is where all tests that use IDC
:6927     Diagnostic Microcode must start from.
:6928     This routine will INIT the IDC to force it to the IDC IDLE LOOP.
:6929     The CSR will then be set up to force the microcode to branch to
:6930     DIAG.IDLE which is the DIAGNOSTIC IDLE LOOP.
:6931     This routine will then clear the DAR which is modified when the IDC
:6932     Microcode branches to DIAG.IDLE.
:6933
:6934     AUTOMODE MODE will then be cleared before returning.
:6935
:6936 .CALLING SEQUENCE:
:6937
:6938     JSR [DIAG.CODE]
:6939
:6940 .INPUT PARAMETERS:
:6941
:6942     NONE
:6943
:6944 .IMPLICIT INPUTS:
:6945
:6946     NONE
:6947
:6948 .OUTPUT PARAMATERS:
:6949
:6950     NONE
:6951
:6952 .IMPLICIT OUTPUTS:
:6953
:6954     AUTOMODE will be cleared.
:6955     The DAR will be cleared.
:6956     The IDC Microcode will be looping at DIAG.IDLE.
:6957
:6958 .SIDE EFFECTS:
:6959
:6960     The IDC CSR will be modified.
:6961
:6962 .---
:6963 *****
:6964 DIAG.CODE:
:6965
:6966     PORT.CONTROL [CLEAR.IDC]                ;SEND IDC MICROCODE TO IDLE
:6967     PORT.CONTROL [CLEAR.IDC]
:6968     PORT.CONTROL [CLEAR.IDC]
:6969     CLR WR[0]
:6970     MOV LS[SETCSR.MAINT] TO WR[0]            ;GET DATA TO SET CSR MAINT BIT
:6971                                           ; CRDY = 0 AND MAINT = 1
:6972     BIS LS[BIT28] TO WR[0]                  ;SET THE TIMEOUT INHIBIT BIT
:6973     PORT.WRITE PORT.ADRS[CSR] WITH WR[0]    ;WRITE TO THE IDC CSR
```

U 072E, 17CC,15
U 072F, 17CC,15
U 0730, 17CC,15
U 0731, 2F80,15
U 0732, 3720,15

U 0733, 4778,15
U 0734, 17A0,15

```

U 0735, B730,15 :6974      MOV LS[#1A] TO WR[0]      ;SET UP A COUNTER TO NOP TO
                  :6975                                     ; ALLOW TIME TO BRANCH FROM
                  :6976                                     ; IDLE TO DIAG.IDLE
                  :6977
                  :6978      NOP.DIAG.CODE:
                  :6979
U 0736, 2100,15 :6980      DEC WR[0]      ; IDC COMPLETES ITS OPERATION
                  :6981      CMP WR[0] WITH LS[ZERO],
U 0737, 4F9C,35 :6982      DT(LONG)&SET.ALU.CC
U 0738, 0873,61 :6983      JMP.IF[NEQ] TO [NOP.DIAG.CODE]
U 0739, 2F80,15 :6984      CLR WR[0]      ;CLEAR THE DAR BECAUSE IT WILL
U 073A, 97A4,15 :6985      PORT.WRITE PORT.ADRS[DAR] WITH WR[0] ; BE INCREMENTED BETWEEN IDLE
                  :6986                                     ; AND DIAG.IDLE
U 073B, 17D4,15 :6987      PORT.CONTROL [CLEAR.AUTO] ;CLEAR AUTOMODE BECAUSE
                  :6988                                     ; AUTOMODE IS NOT CLEARED BY
                  :6989                                     ; INIT
                  :6990
U 073C, 5B00,14 :6991      RETURN
                  :6992
                  :6993
                  :6994
  
```

```

:6995 .PAGE
:6996 .TOC  " WRITE..READ BYTE USING DISKLESS LOOP"
:6997 .+++
:6998
:6999 .FUNCTIONAL DESCRIPTION:
:7000
:7001 .    This routine is used to route a byte of data from FIFO A, thru
:7002 .    the diskless loop and is then written back into FIFO A.
:7003 .    The data pattern is first written into FIFO A address 0.
:7004 .    The CSR is then set up to branch to the IDC diskless loop routine.
:7005 .    The byte of data goes thru the diskless loop and is then loaded
:7006 .    into FIFO A address 1. The expected data is put in WR[1] and
:7007 .    the received data is put in WR[0].
:7008
:7009 .CALLING SEQUENCE:
:7010
:7011 .    JSR [WRITE.READ.BYTE]
:7012
:7013 .INPUT PARAMETERS:
:7014
:7015 .    The write data must be in LS[SAV.DATA.PAT].
:7016 .    WR[2] must contain the drive number of an RL02 or a line with no
:7017 .    disk drive attached, in bits <9:8>.
:7018
:7019 .IMPLICIT INPUTS:
:7020
:7021 .    NONE
:7022
:7023 .OUTPUT PARAMATERS:
:7024
:7025 .    EXPECTED DATA IS IN WR[1]
:7026 .    RECEIVED DATA IS IN WR[0]
:7027
:7028 .IMPLICIT OUTPUTS:
:7029
:7030 .    NONE
:7031
:7032 .SIDE EFFECTS:
:7033
:7034 .    FIFO A will be modified.
:7035 .    The IDC CSR will be modified.
:7036
:7037 .---
:7038 .*****
:7039 .WRITE.READ.BYTE:
:7040 .    PORT.CONTROL [SEL.FIFO.A] ;SELECT FIFO A FOR WRITE
:7041 .    PORT.CONTROL [CLEAR.FIFO.CNTR] ;SET FIFO TO ADDRESS 0
:7042 .    MOV LS[SAV.DATA.PAT] TO WR[0] ;DATA PATT
:7043 .    PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0]
:7044
:7045 .    PORT.CONTROL [SEL.FIFO.A] ;SELECT FIFO A
:7046 .    PORT.CONTROL [CLEAR.FIFO.CNTR] ;SET FIFO TO ADDRESS 0
:7047 .    MOV LS[SETCSR.F4] TO WR[0] ;SET CSR F<2:0>=100 TO BRANCH
:7048 .    ; AND CLR CRDY
:7049 .    BIS WR[2] TO WR[0] ;INCLUDE THE RL02 DRV SELECT
  
```

```

U 073D, 17D8,15 :7040
U 073E, 17C0,15 :7041
U 073F, B724,15 :7042
U 0740, 97A8,15 :7043
:7044
U 0741, 17D8,15 :7045
U 0742, 17C0,15 :7046
U 0743, 3708,15 :7047
:7048
U 0744, AEC4,15 :7049
  
```

U 0745, 17A0,15 :7050 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ; WRITE THE DATA TO IDC CSR

```

:7051
:7052 -----
:7053 -----
:7054 *****
:7055 * IDC *
:7056 *****
:7057 -----
:7058 :018:
:7059 :=000
:7060 :DIAG.IDLE:
:7061
:7062 BEN0/F0,
:7063 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:7064 BEN2/F2,
:7065 NAD/DIAG.IDLE
:7066 -----
:7067 :01C:
:7068 :=100
:7069
:7070 BEN0/CRDY.L,
:7071 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:7072 NAD/TEST.FUNC4 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:7073 -----
:7074 :06B:
:7075 :=0*1
:7076 :DISKLESS.DIAG.0:
:7077
:7078 UMAINT/ON,
:7079 LOOP.LOCK/ON, ;MAINT PREVENTS LOOP LOCK FROM SELECTING
:7080 DISK.WRITE/ON, ;READ DATA...DISK WRITE STILL SELECTS DSRO
:7081 LOAD.LOOP.CNTR.[0E0],
:7082 NAD/DISKLESS.DIAG.1
:7083 -----
:7084 :184:
:7085 :=0
:7086 :DISKLESS.DIAG.1:
:7087
:7088 UMAINT/ON,
:7089 LOOP.LOCK/ON, ;SHIFT OUT 32 ZEROS TO SYNC UP
:7090 DISK.WRITE/ON, ; DATA SEPARATOR
:7091 UCMD/SET.FIFOA,
:7092 INCR.LPCNTR/ON,
:7093 BEN0/CNTR.OVFLW,
:7094 NAD/DISKLESS.DIAG.1
:7095 -----
:7096 :185:
:7097 :=1
:7098 :DISKLESS.DIAG.2:
:7099
:7100 UMAINT/ON,
:7101 LOOP.LOCK/ON,
:7102 DISK.WRITE/ON, ;MUST WAIT 3 CYCLES FOR SYNC SEEN
:7103 ENB.DSK.CLK,
:7104 NAD/DISKLESS.DIAG.3A
    
```

ZZ-ENKCF-1.4
: ENKCF.MCR
: ENKCF.MIC

ENKCF.MIC

WRITE..READ BYTE USING DISKLESS LOOP
MICRO2 1M(01) 15-MAR-83 11:46:22
WRITE..READ BYTE USING DISKLESS LOOP

K 13

15-MAR-83

Fiche 1 Frame K13

Sequence 166

ENKCF Micro-diagnostic for IDC module

Rev 01.40

Page 160

```

:7105 :-----
:7106 :1D9:
:7107 :DISKLESS.DIAG.3A:
:7108 :
:7109 :    UMAINT/ON,
:7110 :    LOOP.LOCK/ON,
:7111 :    DISK.WRITE/ON,
:7112 :    NAD/DISKLESS.DIAG.3B
:7113 :-----
:7114 :1DA:
:7115 :DISKLESS.DIAG.3B:
:7116 :
:7117 :    UMAINT/ON,
:7118 :    LOOP.LOCK/ON,
:7119 :    DISK.WRITE/ON,
:7120 :    NAD/DISKLESS.DIAG.3
:7121 :-----
:7122 :1DB:
:7123 :DISKLESS.DIAG.3:
:7124 :
:7125 :    UMAINT/ON,
:7126 :    LOOP.LOCK/ON,
:7127 :    DISK.WRITE/ON,          ;LOAD THE DATA FROM FIFO A INTO THE DSR
:7128 :    ENB.FIFO/ON,
:7129 :    LOAD.DSR/ON,
:7130 :    NAD/DISKLESS.DIAG.4A
:7131 :-----
:7132 :1DC:
:7133 :DISKLESS.DIAG.4A:
:7134 :
:7135 :    UMAINT/ON,
:7136 :    LOOP.LOCK/ON,          ;INCREMENT THE FIFO ADDRESS COUNTER
:7137 :    DISK.WRITE/ON,
:7138 :    INCR.FIFO.CNTR/ON,
:7139 :    LOAD.LOOP.CNTR.[OF5],
:7140 :    NAD/DISKLESS.DIAG.4
:7141 :-----
:7142 :188:
:7143 :=0
:7144 :DISKLESS.DIAG.4:
:7145 :
:7146 :    UMAINT/ON,
:7147 :    LOOP.LOCK/ON,
:7148 :    DISK.WRITE/ON,          ;WAIT FOR DATA TO SHIFT THRU DISKLESS
:7149 :    INCR.LPCNTR/ON,        ; LOOP
:7150 :    BEN0/CNTR.OVFLW,
:7151 :    NAD/DISKLESS.DIAG.4
:7152 :-----
:7153 :189:
:7154 :=1
:7155 :DISKLESS.DIAG.5:
:7156 :
:7157 :    UMAINT/ON,
:7158 :    LOOP.LOCK/ON,
:7159 :    ENB.DSR/ON,          ;LOAD THE DATA INTO FIFO A
```

ZZ-ENKCF-1.4 :
: ENKCF.MCR
: ENKCF.MIC

ENKCF.MIC

WRITE..READ BYTE US15-MAR-83
MICRO2 1M(01) 15-MAR-83 11:46:22
WRITE..READ BYTE USING DISKLESS LOOP

L 13

Fiche 1 Frame L13

Sequence 167
Rev 01.40 Page 161

ZZ
:
:

```
:7160 : WRT.FIFO/ON,  
:7161 : DISK.WRITE/ON,  
:7162 : NAD/DISKLESS.DIAG.6  
:7163 :-----  
:7164 :1DD:  
:7165 :DISKLESS.DIAG.6:  
:7166 :  
:7167 : UMAINT/ON,  
:7168 : LOOP.LOCK/ON,  
:7169 : DISK.WRITE/ON,  
:7170 : ENB.CPU.CLK, ;SELECT THE CPU CLK  
:7171 : NAD/XFER.REQ ;SEND XFER REQ TO FLAG THE CPU  
:7172 :-----  
:7173 :1D8:  
:7174 :XFER.REQ:  
:7175 :  
:7176 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU  
:7177 : NAD/DIAG.WAIT  
:7178 :-----  
:7179 :1D7:  
:7180 :DIAG.WAIT:  
:7181 :  
:7182 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT  
:7183 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE  
:7184 :-----  
:7185 :159:  
:7186 :=*0*  
:7187 :DIAG.WAIT1:  
:7188 :  
:7189 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND  
:7190 : ; COMMAND FROM THE CPU  
:7191 :-----  
:7192 :15B:  
:7193 :=*1*  
:7194 :DIAG.WAIT2:  
:7195 :  
:7196 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....  
:7197 : ; WAIT SOME MORE  
:7198 :-----  
:7199 :  
:7200 :WAIT.IDC.TSUB:  
U 0746, D800,1A :7201 : SKIP.IF[NO.FAST.INTERRUPT] ;SKIP UNTIL IDC DONE  
U 0747, 8874,94 :7202 : JMP [IDC.DONE.TSUB] ;EXIT LOOP WHEN IDC IS DONE  
U 0748, 8874,64 :7203 : JMP [WAIT.IDC.TSUB] ;WAIT UNTIL IDC IS DONE  
:7204 :  
:7205 :  
:7206 :IDC.DONE.TSUB:  
:7207 :  
U 0749, B700,15 :7208 : MOV LS[SETCSR.F0] TO WR[0] ;CLEAR THE FUNC BITS  
U 074A, 17A0,15 :7209 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ; WRITE THE DATA TO IDC CSR  
U 074B, 90FA,15 :7210 : MISC2 [TRANSFER.GRANT] ;SEND TRANSFER GRANT  
U 074C, 17D8,15 :7211 : PORT.CONTROL [SEL.FIFO.A] ;SEL FIFOA TO READ RECEIVED DATA  
U 074D, 9090,15 :7212 : MISC [SET.PORT.I.0] ;SEL THE PORT TO BUS  
U 074E, 1788,15 :7213 : PORT.READ PORT.ADRS[DATA.BYTE] ;READ THE RETURNED BYTE OF DATA  
U 074F, 3B32,15 :7214 : MOV PORT TO LS[PORT0] ;SAVE THE RECEIVED DATA IN LS
```


ZZ-ENKCF-1.4	:	ENKCF.MIC	WRITE..READ BYTE US15-MAR-83	M 13	Fiche 1	Frame M13	Sequence 168	
:	:	ENKCF.MCR	MICRO2 1M(01) 15-MAR-83 11:46:22	ENKCF Micro-diagnostic for IDC module	Rev 01.40	Page	162	
:	:	ENKCF.MIC	WRITE..READ BYTE USING DISKLESS LOOP					

U 0750,	3732,15	:	7215	MOV LS[PORT0] TO WR[0]	:	SAVE THE DATA IN WR[0] FOR CMP
U 0751,	3724,95	:	7216	MOV LS[SAV.DATA.PAT] TO WR[1]	:	GET THE EXPECTED DATA
U 0752,	5B00,14	:	7217	RETURN	:	RETURN TO MAIN ROUTINE
		:	7218			
		:	7219			
		:	7220			
		:	7221			

```

:7222 .PAGE
:7223 .TOC  " CLEANUP ROUTINE "
:7224 .+++
:7225
:7226 .FUNCTIONAL DESCRIPTION:
:7227
:7228 .      This routine should be used at the end of tests to INIT the
:7229 .      IDC to cleanup any special cinditions that were set by a test
:7230 .      and to JAM to IDC microcode into the IDC IDLE LOOP.
:7231
:7232 .CALLING SEQUENCE:
:7233
:7234 .      JSR [CLEANUP]
:7235
:7236 .INPUT PARAMETERS:
:7237
:7238 .      NONE
:7239
:7240 .IMPLICIT INPUTS:
:7241
:7242 .      NONE
:7243
:7244 .OUTPUT PARAMATERS:
:7245
:7246 .      NONE
:7247
:7248 .IMPLICIT OUTPUTS:
:7249
:7250 .      NONE
:7251
:7252 .SIDE EFFECTS:
:7253
:7254 .      The IDC will be INITED and the IDC microcode will JAM to the
:7255 .      IDC IDLE LOOP.
:7256
:7257 .---
:7258 .*****
:7259 .CLEANUP:
:7260 .      PORT.CONTROL [CLEAR.IDC]          ;INITIALIZE IDC AND SEND MICROCODE TO
:7261 .                                         ; IDLE
:7262 .      PORT.CONTROL [CLEAR.IDC]
:7263 .      PORT.CONTROL [CLEAR.IDC]
:7264 .      RETURN                          ;RETURN TO TEST
:7265
:7266
:7267

```

```

U 0753, 17CC,15
U 0754, 17CC,15
U 0755, 17CC,15
U_0756, 5B00,14

```

```

:7268 .PAGE 'Test 1 Y BUS VERIFICATION TEST (M8388 IDC MODULE OR M8390 CPU MODULE)''
:7269 ++
:7270
:7271 Test Description:
:7272
:7273 This test will insure that the Y bus is fully operational as seen by
:7274 the CPU. The test will move data from local store to a working
:7275 register and then use the Y Bus to move the data back to a temporary
:7276 location in local store.
:7277 The data patterns that will be used are:
:7278 1. AAAAAAAAAA
:7279 2. 55555555
:7280 3. 33333333
:7281 4. 0F0F0F0F
:7282 5. 00FF00FF
:7283 6. 0000FFFF
:7284
:7285
:7286 Assumptions:
:7287
:7288 It is assumed that the CPU has been tested and was found to
:7289 be fully operational without an IDC module installed.
:7290
:7291 Test Steps:
:7292
:7293 1. Set up error mask, error number, and module number in LS
:7294 (for error printouts) and clear previous error number in LS.
:7295 2. MOV data pattern from LS to WR[1].
:7296 3. MOV WR[1] to a temporary LS location.
:7297 4. MOV the temporary LS location to WR[0] and check.
:7298 5. Repeat test steps 1-4 until all data patterns have been tested.
:7299
:7300 Error:
:7301
:7302 ERROR 1 - Y Bus failed.
:7303
:7304 Printout:
:7305 EXPECTED RECEIVED
:7306 WR[0] WR[0]
:7307
:7308 Debug:
:7309
:7310 ERROR 1:
:7311
:7312 If the CPU Micro Diagnostic was run with no errors prior to the
:7313 addition of the IDC module to the system, then the Y Bus is most
:7314 likely being corrupted by the IDC module.
:7315
:7316 If the CPU Micro Diagnostics were not run prior to beginning
:7317 this test then either the CPU or the FPA Module (if one exists) or
:7318 the IDC module could be at fault.
:7319
:7320 --
:7321 *****
:7322 T.1:

```

U 0757, B65E,15

```

MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT 15 IN WR[0] FOR

```

U 0758, 3E80,15	:7323			: CONTROL AND STATUS WORD
	:7324	MOV WR[0] TO LS[CONTROL.STATUS]		: SET BIT 15 IN CONTROL AND
	:7325			: STATUS WORD - BIT 15
	:7326			: INDICATES START OF TEST TO
	:7327			: CONSOLE
U 0759, 10E0,15	:7328	MISC [SET.CP.ATTN]		: ISSUE CPU ATTN TO CONSOLE
	:7329	WAIT.T1.0:		
U 075A, 0875,A4	:7330	JMP [WAIT.T1.0]		: LOOP HERE WAITING FOR CONSOLE
	:7331			: TO RESPOND
U 075B, 65A0,15	:7332	CLR LS[PREVIOUS.ERROR]		: CLEAR PREVIOUS ERROR NUMBER
	:7333			: IN LS
	:7334			
U 075C, 3642,15	:7335	MOV LS[CPU] TO WR[0]		: STORE MODULE CODE IN LS TO
U 075D, C74A,15	:7336	BIS LS[IDC] TO WR[0]		: INDICATE IDC AND CPU
U 075E, 3E8C,15	:7337	MOV WR[0] TO LS[MODULE.NUM]		: MODULES
	:7338			
U 075F, B640,15	:7339	MOV LS[#1] TO WR[0]		: SET WRO = 1
U 0760, BE82,15	:7340	MOV WR[0] TO LS[ERROR.NUMBER]		: SET UP ERROR NUMBER = 1 IN LS
	:7341			
U 0761, E58A,15	:7342	CLR LS[ERROR.MASK]		: SET UP ERROR MASK TO CHECK
	:7343			: ALL BITS
	:7344			
	:7345	LOOP.T1.1:		
U 0762, B698,15	:7346	MOV LS[#####] TO WR[0]		: LOAD THE WORKING REGISTER
U 0763, BE14,15	:7347	MOV WR[0] TO LS[T10]		: MOV THE DATA OVER THE Y BUS
U 0764, 3614,15	:7348	MOV LS[T10] TO WR[0]		: GET THE RECEIVED DATA
U 0765, 3698,95	:7349	MOV LS[#####] TO WR[1]		: GET THE EXPECTED DATA
U 0766, 0869,3C	:7350	JSR [CHECK.RESULT]		: CHECK THE RESULT
U 0767, 8876,24	:7351	JMP [LOOP.T1.1]		: ERROR LOOP
	:7352	LOOP.T1.2:		
U 0768, 369A,15	:7353	MOV LS[#55555555] TO WR[0]		
U 0769, BE14,15	:7354	MOV WR[0] TO LS[T10]		: MOV THE DATA OVER THE Y BUS
U 076A, 3614,15	:7355	MOV LS[T10] TO WR[0]		: GET THE RECEIVED DATA
U 076B, B69A,95	:7356	MOV LS[#55555555] TO WR[1]		: GET THE EXPECTED DATA
U 076C, 0869,3C	:7357	JSR [CHECK.RESULT]		: CHECK THE RESULT
U 076D, 8876,84	:7358	JMP [LOOP.T1.2]		: ERROR LOOP
	:7359	LOOP.T1.3:		
U 076E, B6C4,15	:7360	MOV LS[#33333333] TO WR[0]		
U 076F, BE14,15	:7361	MOV WR[0] TO LS[T10]		: MOV THE DATA OVER THE Y BUS
U 0770, 3614,15	:7362	MOV LS[T10] TO WR[0]		: GET THE RECEIVED DATA
U 0771, 36C4,95	:7363	MOV LS[#33333333] TO WR[1]		: GET THE EXPECTED DATA
U 0772, 0869,3C	:7364	JSR [CHECK.RESULT]		: CHECK THE RESULT
U 0773, 8876,E4	:7365	JMP [LOOP.T1.3]		: ERROR LOOP
	:7366	LOOP.T1.4:		
U 0774, 36C6,15	:7367	MOV LS[#0F0F0F0F] TO WR[0]		
U 0775, BE14,15	:7368	MOV WR[0] TO LS[T10]		: MOV THE DATA OVER THE Y BUS
U 0776, 3614,15	:7369	MOV LS[T10] TO WR[0]		: GET THE RECEIVED DATA
U 0777, B6C6,95	:7370	MOV LS[#0F0F0F0F] TO WR[1]		: GET THE EXPECTED DATA
U 0778, 0869,3C	:7371	JSR [CHECK.RESULT]		: CHECK THE RESULT
U 0779, 0877,44	:7372	JMP [LOOP.T1.4]		: ERROR LOOP
	:7373	LOOP.T1.5:		
U 077A, B6C8,15	:7374	MOV LS[#00FF00FF] TO WR[0]		
U 077B, BE14,15	:7375	MOV WR[0] TO LS[T10]		: MOV THE DATA OVER THE Y BUS
U 077C, 3614,15	:7376	MOV LS[T10] TO WR[0]		: GET THE RECEIVED DATA
U 077D, 36C8,95	:7377	MOV LS[#00FF00FF] TO WR[1]		: GET THE EXPECTED DATA

```

U 077E, 0869,3C :7378      JSR [CHECK.RESULT]      ;CHECK THE RESULT
U 077F, 8877,A4 :7379      JMP [LOOP.T1.5]      ;ERROR LOOP
                   :7380
U 0780, B772,15 :7381      MOV LS[#0000FFFF] TO WR[0]
U 0781, BE14,15 :7382      MOV WR[0] TO LS[T10]      ;MOV THE DATA OVER THE Y BUS
U 0782, 3614,15 :7383      MOV LS[T10] TO WR[0]      ;GET THE RECEIVED DATA
U 0783, 3772,95 :7384      MOV LS[#0000FFFF] TO WR[1] ;GET THE EXPECTED DATA
U 0784, 0869,3C :7385      JSR [CHECK.RESULT]      ;CHECK THE RESULT
U 0785, 8878,04 :7386      JMP [LOOP.T1.6]      ;ERROR LOOP
                   :7387      END.T1:      ;END TEST
                   :7388
                   :7389
                   :7390
                   :7391
                   :7392

```

:7393 :PAGE 'Test 2 IDC SIZING TEST (M8388 IDC MODULE OR M8390 CPU MODULE)'
:7394 :++
:7395 :
:7396 : Test Description:
:7397 :
:7398 : If this diagnostic is being run under APT and an IDC is not found
:7399 : then an error will be reported.
:7400 : This test will size for the IDC Module by writing and then reading
:7401 : Bit <0> of the Disk Address Register (DAR). If this cannot be
:7402 : accomplished successfully, then the message 'IDC NOT PRESENT' will
:7403 : be printed and the IDC microdiagnostic will not be executed. Otherwise
:7404 : 'IDC PRESENT' will be printed and test execution will continue.
:7405 :
:7406 : Assumptions:
:7407 :
:7408 : It is assumed that the CPU and the Y bus are fully operational.
:7409 :
:7410 : Test Steps:
:7411 :
:7412 : 1. Move zero to WR[0].
:7413 : 2. Move #1 to WR [1].
:7414 : 3. Write a one to DAR <0>.
:7415 : 4. Move WR[0] to a temporary LS location to change the data on the
:7416 : Y BUS.
:7417 : 5. Read DAR <0>.
:7418 : 6. Check result. If incorrect data, print message 'IDC NOT PRESENT' and
:7419 : go to Step 12.
:7420 : 7. Write a zero to DAR <0>.
:7421 : 8. Move WR[1] to a temporary LS location to change the data on the
:7422 : Y BUS.
:7423 : 9. Read DAR <0>.
:7424 : 10. Check result.
:7425 : 11. If data is incorrect, print message 'IDC NOT PRESENT'.
:7426 : 12. If APT is present then force an error because IDC was not found.
:7427 : 13. Go to end of section.
:7428 : 14. Else print 'IDC PRESENT' and continue IDC microdiagnostic execution.
:7429 :
:7430 : Errors:
:7431 :
:7432 : ERROR 1 - The diagnostic was being run under APT and no IDC was found.
:7433 :
:7434 : Printout:
:7435 : EXPECTED RECEIVED
:7436 : N/A N/A
:7437 :
:7438 : Debug:
:7439 :
:7440 : If an IDC is not found by this test and one is known to exist, then
:7441 : TEST 4 DISK ADDRESS REGISTER TEST should be run to allow looping on
:7442 : error. A more detailed debug section is also available in TEST 4.
:7443 :
:7444 : If the IDC is not found and one is known to exist then the fault could
:7445 : be in one of the following:
:7446 :
:7447 : 1. The STATE REG, MISC DECODER PAL on the CPU module which generates

```

:7448 : the PORT INSTR H signal.
:7449 : 2. The A,B ADRS GEN PAL on the CPU module which generates the
:7450 : READ PORT L signal.
:7451 : 3. Bit <0> of the data bus transceiver
:7452 : 4. Bit <0> of the IDC I/O bus
:7453 : 5. Bit <0> of the DAR
:7454 : 6. The DAR control signals, WRITE DAR or SEL DAR which are decoded from
:7455 : the Port instruction through the PORT INTERFACE REG PAL, the PORT
:7456 : RD/WR DECODE PAL and the Port instruction decoder.
:7457 :
:7458 :--
:7459 :*****
:7460 :T.2:
U 0786, B65E,15 :7461 :MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT 15 IN WR[0] FOR
:7462 : ;CONTROL AND STATUS WORD
U 0787, 3E80,15 :7463 :MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND
:7464 : ;STATUS WORD - BIT 15
:7465 : ;INDICATES START OF TEST TO
:7466 : ;CONSOLE
U 0788, 10E0,15 :7467 :MISC [SET.CP.ATTN] ;ISSUE CPU ATTN TO CONSOLE
:7468 :WAIT.T2.0:
U 0789, 8878,94 :7469 :JMP [WAIT.T2.0] ;LOOP HERE WAITING FOR CONSOLE
:7470 : ;TO RESPOND
:7471 :
U 078A, E50E,15 :7472 :CLR LS[T7.SUB] ;SET UP FOR MESSAGE PRINTOUT
U 078B, B640,15 :7473 :MOV LS[#1] TO WR[0]
U 078C, BE82,15 :7474 :MOV WR[0] TO LS[ERROR.NUMBER] ;SET ERROR NUMBER =1
:7475 :*****
U 078D, 3776,15 :7476 :MOV LS[DAR.MASK] TO WR[0] ;CHANGE MAR 15,83 PETER GREEN
U 078E, C566,15 :7477 :BIC LS[BIT19] TO WR[0] ;MANUFACT PROB W/SLOW CLOCK
U 078F, 4568,15 :7478 :BIC LS[BIT20] TO WR[0] ;IDC NOT FOUND (INTERMITTANT)
U 0790, 3E8A,15 :7479 :MOV WR[0] TO LS[ERROR.MASK] ;BITS 25 & 27 BEING SET
:7480 : ;SET UP ERROR MASK TO CHECK
:7481 : ;BITS <20:0> IDAR
:7482 :*****
:7483 :LOOP.T2.1:
:7484 :WRITE1.DAR.T2:
U 0791, B640,15 :7485 :MOV LS[#1] TO WR[0] ;GET DATA TO WRITE TO DAR
U 0792, 97A4,15 :7486 :PORT.WRITE PORT.ADRS[DAR] WITH WR[0] ;WRITE A ONE TO IDC DISK
:7487 : ;ADDRESS REGISTER
:7488 :
U 0793, 369C,15 :7489 :MOV LS[ZERO] TO WR[0] ;CHANGE THE DATA IN WR[0]
U 0794, 3F32,95 :7490 :MOV WR[1] TO LS[PORT0] ;AND IN LS[PORT0]
:7491 :
U 0795, 9090,15 :7492 :MISC [SET.PORT.I.O] ;SELECT THE PORT TO BUS
U 0796, 1784,15 :7493 :PORT.READ PORT.ADRS[DAR] ;READ BACK THE IDC DAR
U 0797, 3B32,15 :7494 :MOV PORT TO LS[PORT0] ;MOV THE RECEIVED DATA TO LS
U 0798, 3732,15 :7495 :MOV LS[PORT0] TO WR[0] ;MOV RECEIVED DATA FOR COMPARE
U 0799, 3640,95 :7496 :MOV LS[#1] TO WR[1] ;GET EXPECTED DATA
:7497 :
U 079A, 458A,15 :7498 :BIC LS[ERROR.MASK] TO WR[0] ;CHANGE 3/15 P. G.
:7499 : ;MASK BITS 20-31 FIX
U 079B, C58A,95 :7500 :BIC LS[ERROR.MASK] TO WR[1] ;CLOCK MARGINS
:7501 :
:7502 :CMP WR[0] WITH WR[1], ;CMP THE EXP AND REC DATA

```

G 14

ZZ-ENKCF-1.4 : ENKCF.MIC Test 2 IDC SIZING 15-MAR-83 Fiche 1 Frame G14 Sequence 175
: ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40 Page 169
: ENKCF.MIC Test 2 IDC SIZING TEST (M8388 IDC MODULE OR M8390 CPU MODULE)

```

U 079C, 2640,B5 :7503      DT(LONG)&SET.ALU.CC
U 079D, 887A,B1 :7504      JMP.IF[NEQ] TO [NO.IDC.T2]      ;GO PRINT NO IDC
:7505
:7506      WRITE0.DAR.T2:
U 079E, 369C,15 :7507      MOV LS[#0] TO WR[0]      ;GET DATA TO WRITE TO DAR
U 079F, 97A4,15 :7508      PORT.WRITE PORT.ADRS[DAR] WITH WR[0] ;WRITE A ZERO TO IDC DISK
:7509      ; ADDRESS REGISTER
:7510
U 07A0, B640,15 :7511      MOV LS[#1] TO WR[0]      ;CHANGE THE DATA IN WR[0]
U 07A1, 3F32,95 :7512      MOV WR[1] TO LS[PORT0]      ; AND LS[PORT0]
:7513
U 07A2, 9090,15 :7514      MISC [SET.PORT.I.0]      ;SELECT THE PORT TO BUS
U 07A3, 1784,15 :7515      PORT.READ PORT.ADRS[DAR]      ;READ BACK THE IDC DAR
U 07A4, 3B32,15 :7516      MOV PORT TO LS[PORT0]      ;MOV THE RECEIVED DATA TO LS
U 07A5, 3732,15 :7517      MOV LS[PORT0] TO WR[0]      ;MOV RECEIVED DATA FOR COMPARE
U 07A6, B69C,95 :7518      MOV LS[#0] TO WR[1]      ;GET EXPECTED DATA
:7519
U 07A7, 458A,15 :7520      BIC LS[ERROR.MASK] TO WR[0]      ;CHANGE 3/15 P. G.
:7521      ;MASK BITS 20-31 FIX
U 07A8, C58A,95 :7522      BIC LS[ERROR.MASK] TO WR[1]      ;CLOCK MARGINS
:7523
:7524      CMP WR[0] WITH WR[1],
U 07A9, 2640,B5 :7525      DT(LONG)&SET.ALU.CC      ;CMP THE EXP AND REC DATA
U 07AA, 087B,99 :7526      JMP.IF[EQL] TO [IDC.FOUND.T2]      ;GO PRINT IDC FOUND
:7527
:7528      NO.IDC.T2:
U 07AB, 367E,15 :7529      MOV LS[PRINT.IDC] TO WR[0]      ;SET UP CONTROL STATUS WORD TO
U 07AC, 3E80,15 :7530      MOV WR[0] TO LS[CONTROL.STATUS]      ; PRINT NO IDC FOUND
U 07AD, 10E0,15 :7531      MISC [SET.CP.ATTN]      ;SET ATTENTION TO THE CONSOLE
:7532
:7533      WAIT.T2.1:
U 07AE, 887A,E4 :7534      JMP [WAIT.T2.1]      ;WAIT FOR CONSOLE TO RETURN
:7535      ; CONTROL
:7536
U 07AF, 3690,15 :7537      MOV LS[ERROR.CONTROL] TO WR[0]      ;RUNNING UNDER APT?
:7538      BIT LS[APT.PRESENT] WITH WR[0],
U 07B0, 594A,35 :7539      DT(LONG)&SET.ALU.CC
U 07B1, 09BE,E9 :7540      JMP.IF[BIT.CLR] TO [END.SECTION]      ;IF APT NOT PRESENT THEN DO
:7541      ; NOT RUN DIAGNOSTIC
U 07B2, B66E,15 :7542      MOV LS[NA.EXP.REC] TO WR[0]
U 07B3, 3E80,15 :7543      MOV WR[0] TO LS[CONTROL.STATUS]      ;SET UP TO FORCE AN ERROR
U 07B4, 2F80,15 :7544      CLR WR[0]
U 07B5, 3640,95 :7545      MOV LS[#1] TO WR[1]
U 07B6, 0869,3C :7546      JSR [CHECK.RESULT]      ;GO FORCE THE ERROR
U 07B7, 8879,14 :7547      JMP [LOOP.T2.1]      ;LOOP HERE IF ERR & LOE
:7548
U 07B8, 89BE,E4 :7549      JMP [END.SECTION]      ;NO IDC AVAILABLE, GO TO END
:7550      ; OF SECTION
:7551
:7552      IDC.FOUND.T2:
U 07B9, 7F0E,15 :7553      INC LS[T7.SUB]      ;SET TO PRINT IDC FOUND
U 07BA, 367E,15 :7554      MOV LS[PRINT.IDC] TO WR[0]      ;SET CONTROL STATUS WORD TO
U 07BB, 3E80,15 :7555      MOV WR[0] TO LS[CONTROL.STATUS]      ; PRINT MESSAGE
U 07BC, 10E0,15 :7556      MISC [SET.CP.ATTN]      ;SET ATTENTION TO THE CONSOLE
:7557
:7558      WAIT.T2.2:
U 07BD, 087B,D4 :7559      JMP [WAIT.T2.2]      ;WAIT FOR CONSOLE TO RETURN

```


:7566 .PAGE "Test 3 CRDY & XFER.REQ INIT TEST (M8388 IDC OR M8390 CPU MODULE)"

:7567 :++

:7569 : Test Description:

```

:7571 : This test is designed to insure that the IDC initializes certain
:7572 : important signals properly when a CLEAR.IDC is issued by the
:7573 : CPU. CRDY and XFER.REQ must initialize properly for the IDC
:7574 : microcode branching to work correctly. CRDY should be set
:7575 : and XFER.REQ should be cleared by a CLEAR.IDC instruction.

```

:7577 : Assumptions:

```

:7578 :
:7579 : It is assumed that the CPU and the Y Bus are fully operational.

```

:7581 : Test Steps:

- ```

:7582 :
:7583 :
:7584 :
:7585 :
:7586 :
:7587 :
:7588 :
:7589 :
:7590 :

```
1. Set up error mask, error number and module number in LS (for error printouts) and clear the previous error number in LS.
  2. Issue a CLEAR.IDC.
  3. Allow time for IDC to initialize.
  4. Check CSR bit <7> (CRDY).
  5. Increment the error number.
  6. Check XFER.REQ (FAST INTERRUPT) to be clear.
  7. End test.

```

:7592 : Errors:

```

```

:7593 :
:7594 : ERROR 1 - CSR <7> (CDRY) did not set.

```

```

:7595 : Printout:

```

|       |   |          |          |
|-------|---|----------|----------|
| :7596 | : | EXPECTED | RECEIVED |
| :7597 | : | CSR      | CSR      |

```

:7600 : ERROR 2 - XFER.REQ did not clear when CLEAR.IDC was issued.

```

```

:7600 : ERROR 2 OPEN:REC
:7601 : Printout:

```

| WR[0] | EXPECTED<br>WR[0] | RECEIVED<br>WR[0] |
|-------|-------------------|-------------------|
| 7602  |                   |                   |
| 7603  |                   |                   |

```

;7604 ; Debug:

```

```

:7606 : ERROR 1:

```

```

:7607 :
:7608 : The CLEAR.IDC instruction will be decoded in the PORT RD/WR DECODE
:7609 : PAL. The inputs will be:

```

7610 : CSR 17 H - H

CSR 14 H = H

CSR 12 H = L

CSR 11 H = H

CSR 10 H = H

PORT INSTR H = H

```

:7616 : This combination of inputs will assert CLEAR IDC L. This signal is
:7617 : an input to the INIT Flip Flop. After 2 P2 CLOCKS, INIT H will
:7618 : assert. This signal is inverted to INIT L. INIT L is an input to the
:7619 : CONTROL STATUS REG PAL. At the next P2 CLOCK L, CRDY L will assert.
:7620 : CRDY L is an input to the CSR LINE DRIVERS and will be read when a

```

[illegible]

U 07BE, B65E,15  
U 07BF, 3E80,15

```

U 07C0, 10E0,15 :7676 MISC [SET.CP.ATTN] ; CONSOLE
:7677 WAIT.T3.0: ;ISSUE CPU ATTN TO CONSOLE
U 07C1, 887C,14 :7678 JMP [WAIT.T3.0] ;LOOP HERE WAITING FOR
:7679 ; CONSOLE TO RESPOND
:7680
U 07C2, 65A0,15 :7681 CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
:7682 ; IN LS
:7683
U 07C3, 3642,15 :7684 MOV LS[CPU] TO WR[0] ;STORE MODULE CODE IN LS TO
U 07C4, C74A,15 :7685 BIS LS[IDC] TO WR[0] ; INDICATE IDC AND CPU
U 07C5, 3E8C,15 :7686 MOV WR[0] TO LS[MODULE.NUM] ; MODULES
:7687
:7688
:7689
U 07C6, B640,15 :7690 MOV LS[#1] TO WR[0] ;SET WRO = 1
U 07C7, BE82,15 :7691 MOV WR[0] TO LS[ERROR.NUMBER] ;SET UP ERROR NUMBER = 1 IN LS
:7692
U 07C8, B782,15 :7693 MOV LS[BIT7.MASK] TO WR[0] ;SET MASK TO CHECK ONLY BIT 7
U 07C9, 3E8A,15 :7694 MOV WR[0] TO LS[ERROR.MASK]
:7695
:7696
U 07CA, 17CC,15 :7697 LOOP.T3.1: ;ISSUE INIT TO THE IDC
U 07CB, 17CC,15 :7698 PORT.CONTROL [CLEAR.IDC]
U 07CC, 17CC,15 :7699 PORT.CONTROL [CLEAR.IDC]
U 07CD, 9090,15 :7700 PORT.CONTROL [CLEAR.IDC]
U 07CE, 9780,15 :7701 MISC [SET.PORT.I.0] ;SELECT THE IDC
U 07CF, 3B32,15 :7702 PORT.READ PORT.ADRS[CSR] ;READ THE CSR
U 07D0, 3732,15 :7703 MOV PORT TO LS[PORT0] ;SAVE THE DATA IN LS
U 07D1, B64E,95 :7704 MOV LS[PORT0] TO WR[0] ;SAVE THE RECEIVED DATA
U 07D2, 0869,3C :7705 MOV LS[BIT7] TO WR[1] ;SET THE EXPECTED DATA
U 07D3, 087C,A4 :7706 JSR [CHECK.RESULT] ;COMPARE THE DATA
:7707 JMP [LOOP.T3.1] ;LOOP HERE IF ERR & LOE IS SET
:7708
:7709 CHECK.XFER.REQ.T3:
:7710 LOOP.T3.2:
U 07D4, 3642,15 :7711 MOV LS[#2] TO WR[0] ;SET UP FOR ERROR 2
U 07D5, BE82,15 :7712 MOV WR[0] TO LS[ERROR.NUMBER]
U 07D6, E58A,15 :7713 CLR LS[ERROR.MASK] ;SET TO CHECK ALL BITS
U 07D7, 17CC,15 :7714 PORT.CONTROL [CLEAR.IDC] ;ISSUE INIT TO THE IDC
U 07D8, 17CC,15 :7715 PORT.CONTROL [CLEAR.IDC]
U 07D9, 17CC,15 :7716 PORT.CONTROL [CLEAR.IDC]
U 07DA, 2F80,15 :7717 CLR WR[0] ;CLR WR 0 FOR FUTURE USE
U 07DB, DB00,15 :7718 NOP ;ALLOW TIME FOR XFER REQ
U 07DC, DB00,15 :7719 NOP ; TO BE RECOGNIZED BY THE
U 07DD, DB00,15 :7720 NOP ; CPU
U 07DE, DB00,1A :7721 SKIP.IF[NO.FAST.INTERRUPT]
U 07DF, 2040,15 :7722 INC WR[0] ;INCR WR 0 IF FAST INTERRUPT
:7723 ; IS ASSERTED
U 07E0, 2F82,95 :7724 CLR WR[1] ;GET THE EXPECTED DATA
U 07E1, 0869,3C :7725 JSR [CHECK.RESULT] ;COMPARE THE DATA
U 07E2, 087D,44 :7726 JMP [LOOP.T3.2] ;LOOP HERE IF ERR & LOE IS SET
U 07E3, 8875,3C :7727 JSR [CLEANUP] ;CLEANUP CONDITIONS SET BY THIS
:7728 ; TEST
:7729 ;END TEST
:7730 END.T3:

```

ZZ-ENKCF-1.4 : ENKCF.MIC Test 3 CRDY & XFER15-MAR-83 L 14 Fiche 1 Frame L14 Sequence 180  
: ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40 Page 174  
: ENKCF.MIC Test 3 CRDY & XFER.REQ INIT TEST (M8388 IDC OR M8390 CPU MODULE)

:7731  
:7732  
:7733

ZZ  
:  
:

```

:7734 .PAGE "Test 4 DISK ADDRESS REGISTER TEST (M8388 IDC MODULE)"
:7735 .++
:7736
:7737 .Test Description:
:7738
:7739 .This test is designed to verify the integrity of the Disk Address
:7740 .Register (DAR).
:7741 .The following patterns will be used:
:7742 .1. 2AAAA
:7743 .2. 55555
:7744 .3. 33333
:7745 .4. 70F0F
:7746 .5. 700FF
:7747 .6. 0FFFF
:7748
:7749 .The DAR will then be written with 7FFFF. A CLEAR.IDC will then be
:7750 .be issued and the DAR checked to insure that all bits cleared.
:7751
:7752 .Assumptions:
:7753
:7754 .It is assumed that the CPU and the Y Bus are fully operational.
:7755
:7756 .Test Steps:
:7757
:7758 .1. Set up error mask, error number and module number in LS
:7759 .(for error printouts) and clear the previous error number in LS.
:7760 .2. CLEAR.IDC to force the IDC microcode to IDLE.
:7761 .3. Move the data pattern from LS to WR 0.
:7762 .4. Write the data pattern to the DAR.
:7763 .5. Read the DAR.
:7764 .6. Check the data.
:7765 .7. Get next data pattern and GO TO Step 2 until all data patterns
:7766 .have been tested.
:7767 .8. Increment the error number.
:7768 .9. Write the DAR with 7FFFF.
:7769 .10. Issue a CLEAR.IDC and wait for initialize.
:7770 .11. Check that the DAR cleared.
:7771 .12. End test.
:7772
:7773 .Errors:
:7774
:7775 .ERROR 1 - The DAR failed.
:7776
:7777 .Printout:
:7778 .EXPECTED RECEIVED
:7779 .DAR DAR
:7780
:7781 .ERROR 2 - The DAR did not clear on a CLEAR.IDC.
:7782 .Printout:
:7783 .EXPECTED RECEIVED
:7784 .DAR DAR
:7785
:7786 .Debug:
:7787 .ERROR 1:
:7788

```

```

:7789 : WRITE.DAR:
:7790 : To allow the data pattern to be written into the DAR, the inputs to
:7791 : the PORT RD/WR DECODE PAL will be:
:7792 : CSR 17 H = H
:7793 : CSR 14 H = L
:7794 : CSR 13 H = H
:7795 : CSR 12 H = L
:7796 : CSR 11 H = L
:7797 : CSR 10 H = H
:7798 : PORT INSTR H = H
:7799 : These inputs will assert WRITE DAR L = L, which will load the
:7800 : DAR on the next Current Clk pulse with the data which is on the I/O
:7801 : BUS.
:7802 :
:7803 : READ.DAR:
:7804 : To read the DAR, the PORT INTERFACE REG PAL and the Port Instruction
:7805 : Decoder must operate correctly.
:7806 : The CPU first issues a MISC [SET.PORT.I.O] which will assert
:7807 : SEL IDC L.
:7808 : The inputs to the PORT INTERFACE REGISTER PAL on the
:7809 : PORT.READ PORT.ADRS[DAR] instruction will be as follows:
:7810 : CSR 17 H = H
:7811 : CSR 14 H = L
:7812 : CSR 12 H = L
:7813 : CSR 11 H = L
:7814 : CSR 10 H = H
:7815 : PORT INSTR H = H
:7816 : CPU P2 H = H
:7817 : The resultant outputs will be:
:7818 : R2 L = H
:7819 : R1 L = H
:7820 : R0 L = L
:7821 : The CPU then issues a MOV PORT TO LSC[] which asserts READ PORT L which
:7822 : is an input to the PORT INTERFACE REG PAL. This input will cause
:7823 : READ PORT H to assert. READ PORT H is ANDED with SEL IDC H = H and
:7824 : SEL ACC L = H and will assert READ IDC L.
:7825 : READ IDC L and READ PORT H are the enables for the decoder of R<2:0>.
:7826 : R<2:0> will be decoded and will assert SEL DAR L which enables the
:7827 : DAR onto the I/O Bus. READ IDC L also changes the direction of the
:7828 : BUS TRANCEIVERS thus enabling the data from the I/O BUS onto the
:7829 : Y BUS.
:7830 :
:7831 : ERROR 2:
:7832 :
:7833 : The CLEAR.IDC instruction will be decoded in the PORT RD/WR DECODE
:7834 : PAL. The inputs will be:
:7835 : CSR 17 H = H
:7836 : CSR 14 H = H
:7837 : CSR 12 H = L
:7838 : CSR 11 H = H
:7839 : CSR 10 H = H
:7840 : PORT INSTR H = H
:7841 : This combination of inputs will assert CLEAR IDC L. This signal is
:7842 : an input to the INIT Flip Flop. After 2 P2 CLOCKS, INIT H will
:7843 : assert. INIT H is the RESET input to the Disk Address Register and

```

U (

U 1

U (

41

44

111

U (

U (

U 1

U (

44

114

04

U (

U (

U (

U 1

U 1

U (

U (

116

U (

```

:7844 : when asserted will clear all bits in the DAR.
:7845 :
:7846 :--
:7847 :*****
:7848 :
U 07E4, B65E,15 :7849 T.4: MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT 15 IN WR[0] FOR
:7850 : CONTROL AND STATUS WORD
U 07E5, 3E80,15 :7851 MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND
:7852 : STATUS WORD - BIT 15
:7853 : INDICATES START OF TEST TO
:7854 : CONSOLE
U 07E6, 10E0,15 :7855 MISC [SET.CP.ATTN] ;ISSUE CPU ATTN TO CONSOLE
:7856 WAIT.T4.0:
U 07E7, 087E,74 :7857 JMP [WAIT.T4.0] ;LOOP HERE WAITING FOR
:7858 : CONSOLE TO RESPOND
:7859
U 07E8, 65A0,15 :7860 CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
:7861 : IN LS
:7862
U 07E9, B64A,15 :7863 MOV LS[IDC] TO WR[0] ;STORE MODULE CODE IN LS TO
U 07EA, 3E8C,15 :7864 MOV WR[0] TO LS[MODULE.NUM] ;INDICATE IDC MODULE
:7865
U 07EB, B640,15 :7866 MOV LS[#1] TO WR[0] ;SET WRO = 1
U 07EC, BE82,15 :7867 MOV WR[0] TO LS[ERROR.NUMBER] ;SET UP ERROR NUMBER = 1 IN LS
:7868
U 07ED, 3776,15 :7869 MOV LS[DAR.MASK] TO WR[0] ;GET THE ERROR MASK
U 07EE, 3E8A,15 :7870 MOV WR[0] TO LS[ERROR.MASK] ;SET UP ERROR MASK TO CHECK
:7871 : BITS <18:0> DAR
:7872
U 07EF, 17CC,15 :7873 PORT.CONTROL [CLEAR.IDC] ;FORCE THE IDC MICROCODE TO
:7874 : THE IDLE LOOP
U 07F0, 17CC,15 :7875 PORT.CONTROL [CLEAR.IDC]
U 07F1, 17CC,15 :7876 PORT.CONTROL [CLEAR.IDC]
:7877 WRITE.DAR.T4.1:
:7878 LOOP.T4.1:
U 07F2, B698,15 :7879 MOV LS[#AAAAAAAA] TO WR[0] ;GET THE DATA PATTERN TO WRITE
U 07F3, B776,95 :7880 MOV LS[#FFF80000] TO WR[1] ;MASK FOR UPPER BITS
U 07F4, AF42,15 :7881 BIC WR[1] TO WR[0] ;MASK OFF BITS <31:19>
U 07F5, 3F24,15 :7882 MOV WR[0] TO LS[SAV.DATA.PAT] ;SAVE THE EXPECTED DATA
U 07F6, 97A4,15 :7883 PORT.WRITE PORT.ADRS[DAR] WITH WR[0] ;WRITE THE DATA PAT TO THE DAR
:7884
:7885 READ.DAR.T4.1:
U 07F7, 9090,15 :7886 MISC [SET.PORT.I.0] ;SELECT THE IDC
U 07F8, 1784,15 :7887 PORT.READ PORT.ADRS[DAR] ;SET UP TO READ THE DAR
U 07F9, 3B32,15 :7888 MOV PORT TO LS[PORT0] ;READ THE DATA FROM DAR AND
:7889 : SAVE IT IN LS
U 07FA, 3732,15 :7890 MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
U 07FB, 3724,95 :7891 MOV LS[SAV.DATA.PAT] TO WR[1] ;SET UP EXPECTED DATA
U 07FC, 0869,3C :7892 JSR [CHECK.RESULT] ;CHECK THE DATA
U 07FD, 887F,24 :7893 JMP [LOOP.T4.1] ;LOOP HERE IF ERR & LOE IS SET
:7894
:7895 WRITE.DAR.T4.2:
:7896 LOOP.T4.2:
U 07FE, 369A,15 :7897 MOV LS[#55555555] TO WR[0] ;GET THE DATA PATTERN TO WRITE
U 07FF, B776,95 :7898 MOV LS[#FFF80000] TO WR[1] ;MASK FOR UPPER BITS

```



C 15

ZZ-ENKCF-1.4 : ENKCF.MIC Test 4 DISK ADDRESS REGISTER TEST (M8388 IDC MODULE) Fiche 1 Frame C15 Sequence 184  
 : ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40 Page 178  
 : ENKCF.MIC Test 4 DISK ADDRESS REGISTER TEST (M8388 IDC MODULE)

```

U 0800, AF42,15 :7899 BIC WR[1] TO WR[0] ;MASK OFF BITS <31:19>
U 0801, 3F24,15 :7900 MOV WR[0] TO LS[SAV.DATA.PAT] ;SAVE THE EXPECTED DATA
U 0802, 97A4,15 :7901 PORT.WRITE PORT.ADRS[DAR] WITH WR[0] ;WRITE THE DATA PAT TO THE DAR
 :7902
 :7903
U 0803, 9090,15 :7904 READ.DAR.T4.2:
U 0804, 1784,15 :7905 MISC [SET.PORT.I.0] ;SELECT THE IDC
U 0805, 3B32,15 :7906 PORT.READ PORT.ADRS[DAR] ;SET UP TO READ THE DAR
 :7907 MOV PORT TO LS[PORT0] ;READ THE DATA FROM DAR AND
 :7908 MOV LS[PORT0] TO WR[0] ;SAVE IT IN LS
U 0806, 3732,15 :7909 MOV LS[SAV.DATA.PAT] TO WR[1] ;SET UP RECEIVED DATA
U 0807, 3724,95 :7910 JSR [CHECK.RESULT] ;SET UP EXPECTED DATA
U 0808, 0869,3C :7911 JMP [LOOP.T4.2] ;CHECK THE DATA
U 0809, 887F,E4 :7912 ;LOOP HERE IF ERR & LOE IS SET
 :7913
 :7914
U 080A, B6C4,15 :7915 WRITE.DAR.T4.3:
U 080B, B776,95 :7916 LOOP.T4.3:
U 080C, AF42,15 :7917 MOV LS[#33333333] TO WR[0] ;GET THE DATA PATTERN TO WRITE
U 080D, 3F24,15 :7918 MOV LS[#FFF80000] TO WR[1] ;MASK FOR UPPER BITS
U 080E, 97A4,15 :7919 BIC WR[1] TO WR[0] ;MASK OFF BITS <31:19>
 :7920 MOV WR[0] TO LS[SAV.DATA.PAT] ;SAVE THE EXPECTED DATA
 :7921 PORT.WRITE PORT.ADRS[DAR] WITH WR[0] ;WRITE THE DATA PAT TO THE DAR
U 080F, 9090,15 :7922 READ.DAR.T4.3:
U 0810, 1784,15 :7923 MISC [SET.PORT.I.0] ;SELECT THE IDC
U 0811, 3B32,15 :7924 PORT.READ PORT.ADRS[DAR] ;SET UP TO READ THE DAR
 :7925 MOV PORT TO LS[PORT0] ;READ THE DATA FROM DAR AND
 :7926 MOV LS[PORT0] TO WR[0] ;SAVE IT IN LS
U 0812, 3732,15 :7927 MOV LS[SAV.DATA.PAT] TO WR[1] ;SET UP RECEIVED DATA
U 0813, 3724,95 :7928 JSR [CHECK.RESULT] ;SET UP EXPECTED DATA
U 0814, 0869,3C :7929 JMP [LOOP.T4.3] ;CHECK THE DATA
U 0815, 0880,A4 :7930 ;LOOP HERE IF ERR & LOE IS SET
 :7931
 :7932
U 0816, 36C6,15 :7933 WRITE.DAR.T4.4:
U 0817, B776,95 :7934 LOOP.T4.4:
U 0818, AF42,15 :7935 MOV LS[#0F0F0F0F] TO WR[0] ;GET THE DATA PATTERN TO WRITE
U 0819, 3F24,15 :7936 MOV LS[#FFF80000] TO WR[1] ;MASK FOR UPPER BITS
U 081A, 97A4,15 :7937 BIC WR[1] TO WR[0] ;MASK OFF BITS <31:19>
 :7938 MOV WR[0] TO LS[SAV.DATA.PAT] ;SAVE THE EXPECTED DATA
 :7939 PORT.WRITE PORT.ADRS[DAR] WITH WR[0] ;WRITE THE DATA PAT TO THE DAR
U 081B, 9090,15 :7940 READ.DAR.T4.4:
U 081C, 1784,15 :7941 MISC [SET.PORT.I.0] ;SELECT THE IDC
U 081D, 3B32,15 :7942 PORT.READ PORT.ADRS[DAR] ;SET UP TO READ THE DAR
 :7943 MOV PORT TO LS[PORT0] ;READ THE DATA FROM DAR AND
 :7944 MOV LS[PORT0] TO WR[0] ;SAVE IT IN LS
U 081E, 3732,15 :7945 MOV LS[SAV.DATA.PAT] TO WR[1] ;SET UP RECEIVED DATA
U 081F, 3724,95 :7946 JSR [CHECK.RESULT] ;SET UP EXPECTED DATA
U 0820, 0869,3C :7947 JMP [LOOP.T4.4] ;CHECK THE DATA
U 0821, 8881,64 :7948 ;LOOP HERE IF ERR & LOE IS SET
 :7949
 :7950
U 0822, B6C8,15 :7951 WRITE.DAR.T4.5:
U 0823, B776,95 :7952 LOOP.T4.5:
U 0824, AF42,15 :7953 MOV LS[#00FF00FF] TO WR[0] ;GET THE DATA PATTERN TO WRITE
 :7954 MOV LS[#FFF80000] TO WR[1] ;MASK FOR UPPER BITS
 :7955 BIC WR[1] TO WR[0] ;MASK OFF BITS <31:19>

```

D 15

|              |           |                                                      |                    |                                       |              |
|--------------|-----------|------------------------------------------------------|--------------------|---------------------------------------|--------------|
| ZZ-ENKCF-1.4 | ENKCF.MIC | Test 4 DISK ADDRESS                                  | 15-MAR-83          | Fiche 1 Frame D15                     | Sequence 185 |
| :            | ENKCF.MCR | MICRO2 1M(01)                                        | 15-MAR-83 11:46:22 | ENKCF Micro-diagnostic for IDC module | Rev 01.40    |
| :            | ENKCF.MIC | Test 4 DISK ADDRESS REGISTER TEST (M8388 IDC MODULE) |                    |                                       | Page 179     |

```

U 0825, 3F24,15 :7954 MOV WR[0] TO LS[SAV.DATA.PAT] ;SAVE THE EXPECTED DATA
U 0826, 97A4,15 :7955 PORT.WRITE PORT.ADRS[DAR] WITH WR[0] ;WRITE THE DATA PAT TO THE DAR
 :7956
 :7957
U 0827, 9090,15 :7958 READ.DAR.T4.5:
U 0828, 1784,15 :7959 MISC [SET.PORT.I.0] ;SELECT THE IDC
U 0829, 3B32,15 :7960 PORT.READ PORT.ADRS[DAR] ;SET UP TO READ THE DAR
 :7961 MOV PORT TO LS[PORT0] ;READ THE DATA FROM DAR AND
 :7962 MOV LS[PORT0] TO WR[0] ;SAVE IT IN LS
U 082A, 3732,15 :7963 MOV LS[SAV.DATA.PAT] TO WR[1] ;SET UP RECEIVED DATA
U 082B, 3724,95 :7964 JSR [CHECK.RESULT] ;SET UP EXPECTED DATA
U 082C, 0869,3C :7965 JMP [LOOP.T4.5] ;CHECK THE DATA
U 082D, 0882,24 :7966 ;LOOP HERE IF ERR & LOE IS SET
 :7967
 :7968 WRITE.DAR.T4.6:
U 082E, B772,15 :7969 LOOP.T4.6:
U 082F, B776,95 :7970 MOV LS[#0000FFFF] TO WR[0] ;GET THE DATA PATTERN TO WRITE
U 0830, AF42,15 :7971 MOV LS[#FFF80000] TO WR[1] ;MASK FOR UPPER BITS
U 0831, 3F24,15 :7972 BIC WR[1] TO WR[0] ;MASK OFF BITS <31:19>
U 0832, 97A4,15 :7973 MOV WR[0] TO LS[SAV.DATA.PAT] ;SAVE THE EXPECTED DATA
 :7974 PORT.WRITE PORT.ADRS[DAR] WITH WR[0] ;WRITE THE DATA PAT TO THE DAR
 :7975
U 0833, 9090,15 :7976 READ.DAR.T4.6:
U 0834, 1784,15 :7977 MISC [SET.PORT.I.0] ;SELECT THE IDC
U 0835, 3B32,15 :7978 PORT.READ PORT.ADRS[DAR] ;SET UP TO READ THE DAR
 :7979 MOV PORT TO LS[PORT0] ;READ THE DATA FROM DAR AND
 :7980 MOV LS[PORT0] TO WR[0] ;SAVE IT IN LS
U 0836, 3732,15 :7981 MOV LS[SAV.DATA.PAT] TO WR[1] ;SET UP RECEIVED DATA
U 0837, 3724,95 :7982 JSR [CHECK.RESULT] ;SET UP EXPECTED DATA
U 0838, 0869,3C :7983 JMP [LOOP.T4.6] ;CHECK THE DATA
U 0839, 0882,E4 :7984 ;LOOP HERE IF ERR & LOE IS SET
 :7985
 :7986 CHECK.CLEAR.IDC:
U 083A, FF82,15 :7987 INC LS[ERROR.NUMBER] ;ERROR 2
 :7988
 :7989 LOOP.T4.7:
U 083B, B69E,15 :7990 MOV LS[#FFFFFFFF] TO WR[0] ;GET ALL ONES TO WRITE
U 083C, 97A4,15 :7991 PORT.WRITE PORT.ADRS[DAR] WITH WR[0] ;WRITE TO THE DAR
U 083D, 17CC,15 :7992 PORT.CONTROL [CLEAR.IDC] ;ISSUE INIT TO THE IDC
U 083E, 17CC,15 :7993 PORT.CONTROL [CLEAR.IDC]
U 083F, 17CC,15 :7994 PORT.CONTROL [CLEAR.IDC]
U 0840, 9090,15 :7995 MISC [SET.PORT.I.0] ;SELECT THE IDC
U 0841, 1784,15 :7996 PORT.READ PORT.ADRS[DAR] ;SET UP TO READ THE DAR
 :7997 MOV PORT TO LS[PORT0] ;READ THE DATA FROM DAR AND
 :7998 MOV LS[PORT0] TO WR[0] ;SAVE IT IN LS
U 0843, 3732,15 :7999 CLR WR[1] ;SET UP RECEIVED DATA
U 0844, 2F82,95 :8000 JSR [CHECK.RESULT] ;SET UP THE EXPECTED DATA
U 0845, 0869,3C :8001 JMP [LOOP.T4.7] ;COMPARE THE DATA
U 0846, 8883,B4 :8002 JSR [CLEANUP] ;LOOP HERE IF ERR & LOE IS SET
U 0847, 8875,3C :8003 ;CLEAN UP CONDITIONS SET BY
 :8004 ; THIS TEST
 :8005 ;END TEST
 :8006 END.T4:

```

U  
U  
U  
U  
U  
U  
U

:8062 :  
:8063 : ERROR 1 - The CSR failed.  
:8064 :  
:8065 : ERROR 2 - CSR bit <22> did not clear when a CLEAR.IDC was executed.  
:8066 :  
:8067 : ERROR 3 - CSR bit <23> did not clear when a CLEAR.IDC was executed.  
:8068 :  
:8069 : Debug:  
:8070 :  
:8071 : ERROR 1:  
:8072 :  
:8073 : WRITE.CSR:  
:8074 : To allow the data pattern to be written into the CSR, the inputs to  
:8075 : the PORT RD/WR DECODE PAL will be:  
:8076 : CSR 17 H = H  
:8077 : CSR 14 H = L  
:8078 : CSR 13 H = H  
:8079 : CSR 12 H = L  
:8080 : CSR 11 H = L  
:8081 : CSR 10 H = L  
:8082 : PORT INSTR H = H  
:8083 : These inputs will assert WRITE CSR L = L, which will load the  
:8084 : CSR on the next P2 CLOCK pulse with the data which is on the I/O Bus.  
:8085 :  
:8086 : READ.CSR:  
:8087 : To read the CSR, the PORT INTERFACE REG PAL and the Port Instruction  
:8088 : Decoder must operate correctly.  
:8089 : The CPU first issues a MISC [SET.PORT.I.O] which will assert  
:8090 : SEL IDC L.  
:8091 : The inputs to the PORT INTERFACE REGISTER PAL on the  
:8092 : PORT.READ PORT.ADRS[CSR] instruction will be as follows:  
:8093 : CSR 17 H = H  
:8094 : CSR 14 H = L  
:8095 : CSR 12 H = L  
:8096 : CSR 11 H = L  
:8097 : CSR 10 H = L  
:8098 : PORT INSTR H = H  
:8099 : CPU P2 H = H  
:8100 : The resultant outputs will be:  
:8101 : R2 L = H  
:8102 : R1 L = H  
:8103 : R0 L = H  
:8104 : The CPU then issues a MOV PORT TO LS[] which asserts READ PORT L which  
:8105 : is an input to the PORT INTERFACE REG PAL. This input will cause  
:8106 : READ PORT H to assert. READ PORT H is ANDED with SEL IDC H = H and  
:8107 : SEL ACC L = H and will assert READ IDC L.  
:8108 : READ IDC L and READ PORT H are the enables for the decoder of R<2:0>.  
:8109 : R<2:0> will be decoded and will assert SEL CSR L which enables the  
:8110 : CSR onto the I/O Bus. READ IDC L also changes the direction of the  
:8111 : BUS TRANCEIVERS thus enabling the data from the I/O BUS onto the  
:8112 : Y BUS.  
:8113 :  
:8114 : ERROR 2:  
:8115 :  
:8116 : The CLEAR.IDC instruction will be decoded in the PORT RD/WR DECODE

[illegible]

|              |   |           |        |                                                 |           |          |                                       |              |          |    |
|--------------|---|-----------|--------|-------------------------------------------------|-----------|----------|---------------------------------------|--------------|----------|----|
| ZZ-ENKCF-1.4 | : | ENKCF.MIC | Test 5 | CONTROL STA15-MAR-83                            | H 15      | Fiche 1  | Frame H15                             | Sequence 189 | Page 183 | ZZ |
| :            | : | ENKCF.MCR | MICRO2 | 1M(01)                                          | 15-MAR-83 | 11:46:22 | ENKCF Micro-diagnostic for IDC module | Rev 01.40    |          | :  |
| :            | : | ENKCF.MIC | Test 5 | CONTROL STATUS REGISTER TEST (M8388 IDC MODULE) |           |          |                                       |              |          | :  |

  

|                 |   |      |                                      |                                |  |
|-----------------|---|------|--------------------------------------|--------------------------------|--|
| U 0854, 17CC,15 | : | 8172 | PORT.CONTROL [CLEAR.IDC]             |                                |  |
| U 0855, 17CC,15 | : | 8173 | PORT.CONTROL [CLEAR.IDC]             |                                |  |
|                 | : | 8174 | LOOP.T5.1:                           |                                |  |
| U 0856, B778,15 | : | 8175 | MOV LS[CSR.PAT.1] TO WR[0]           | ;GET THE DATA PATTERN          |  |
|                 | : | 8176 |                                      |                                |  |
|                 | : | 8177 | WRITE.CSR.T5.1:                      |                                |  |
| U 0857, 17A0,15 | : | 8178 | PORT.WRITE PORT.ADRS[CSR] WITH WR[0] | ;WRITE THE DATA TO THE CSR     |  |
|                 | : | 8179 |                                      |                                |  |
|                 | : | 8180 | READ.CSR.T5.1:                       |                                |  |
| U 0858, 9090,15 | : | 8181 | MISC [SET.PORT.1.0]                  | ;SELECT THE IDC                |  |
| U 0859, 9780,15 | : | 8182 | PORT.READ PORT.ADRS[CSR]             | ;SET UP TO READ THE CSR        |  |
| U 085A, 3B32,15 | : | 8183 | MOV PORT TO LS[PORT0]                | ;READ THE DATA FROM CSR AND    |  |
|                 | : | 8184 |                                      | ;SAVE IT IN LS                 |  |
| U 085B, 3732,15 | : | 8185 | MOV LS[PORT0] TO WR[0]               | ;SET UP RECEIVED DATA          |  |
| U 085C, 3778,95 | : | 8186 | MOV LS[CSR.PAT.1] TO WR[1]           | ;SET UP EXPECTED DATA          |  |
| U 085D, 0869,3C | : | 8187 | JSR [CHECK.RESULT]                   | ;CHECK THE DATA                |  |
| U 085E, 0885,64 | : | 8188 | JMP [LOOP.T5.1]                      | ;LOOP HERE IF ERR & LOE IS SET |  |
|                 | : | 8189 |                                      |                                |  |
|                 | : | 8190 | LOOP.T5.2:                           |                                |  |
| U 085F, 377A,15 | : | 8191 | MOV LS[CSR.PAT.2] TO WR[0]           | ;GET THE DATA PATTERN          |  |
|                 | : | 8192 |                                      |                                |  |
|                 | : | 8193 | WRITE.CSR.T5.2:                      |                                |  |
| U 0860, 17A0,15 | : | 8194 | PORT.WRITE PORT.ADRS[CSR] WITH WR[0] | ;WRITE THE DATA TO THE CSR     |  |
|                 | : | 8195 |                                      |                                |  |
|                 | : | 8196 | READ.CSR.T5.2:                       |                                |  |
| U 0861, 9090,15 | : | 8197 | MISC [SET.PORT.1.0]                  | ;SELECT THE IDC                |  |
| U 0862, 9780,15 | : | 8198 | PORT.READ PORT.ADRS[CSR]             | ;SET UP TO READ THE CSR        |  |
| U 0863, 3B32,15 | : | 8199 | MOV PORT TO LS[PORT0]                | ;READ THE DATA FROM CSR AND    |  |
|                 | : | 8200 |                                      | ;SAVE IT IN LS                 |  |
| U 0864, 3732,15 | : | 8201 | MOV LS[PORT0] TO WR[0]               | ;SET UP RECEIVED DATA          |  |
| U 0865, B77A,95 | : | 8202 | MOV LS[CSR.PAT.2] TO WR[1]           | ;SET UP EXPECTED DATA          |  |
| U 0866, 0869,3C | : | 8203 | JSR [CHECK.RESULT]                   | ;CHECK THE DATA                |  |
| U 0867, 0885,F4 | : | 8204 | JMP [LOOP.T5.2]                      | ;LOOP HERE IF ERR & LOE IS SET |  |
|                 | : | 8205 |                                      |                                |  |
|                 | : | 8206 | LOOP.T5.3:                           |                                |  |
| U 0868, 377C,15 | : | 8207 | MOV LS[CSR.PAT.3] TO WR[0]           | ;GET THE DATA PATTERN          |  |
|                 | : | 8208 |                                      |                                |  |
|                 | : | 8209 | WRITE.CSR.T5.3:                      |                                |  |
| U 0869, 17A0,15 | : | 8210 | PORT.WRITE PORT.ADRS[CSR] WITH WR[0] | ;WRITE THE DATA TO THE CSR     |  |
|                 | : | 8211 |                                      |                                |  |
|                 | : | 8212 | READ.CSR.T5.3:                       |                                |  |
| U 086A, 9090,15 | : | 8213 | MISC [SET.PORT.1.0]                  | ;SELECT THE IDC                |  |
| U 086B, 9780,15 | : | 8214 | PORT.READ PORT.ADRS[CSR]             | ;SET UP TO READ THE CSR        |  |
| U 086C, 3B32,15 | : | 8215 | MOV PORT TO LS[PORT0]                | ;READ THE DATA FROM CSR AND    |  |
|                 | : | 8216 |                                      | ;SAVE IT IN LS                 |  |
| U 086D, 3732,15 | : | 8217 | MOV LS[PORT0] TO WR[0]               | ;SET UP RECEIVED DATA          |  |
| U 086E, B77C,95 | : | 8218 | MOV LS[CSR.PAT.3] TO WR[1]           | ;SET UP EXPECTED DATA          |  |
| U 086F, 0869,3C | : | 8219 | JSR [CHECK.RESULT]                   | ;CHECK THE DATA                |  |
| U 0870, 8886,84 | : | 8220 | JMP [LOOP.T5.3]                      | ;LOOP HERE IF ERR & LOE IS SET |  |
|                 | : | 8221 |                                      |                                |  |
|                 | : | 8222 | LOOP.T5.4:                           |                                |  |
| U 0871, B77E,15 | : | 8223 | MOV LS[CSR.PAT.4] TO WR[0]           | ;GET THE DATA PATTERN          |  |
|                 | : | 8224 |                                      |                                |  |
|                 | : | 8225 | WRITE.CSR.T5.4:                      |                                |  |
| U 0872, 17A0,15 | : | 8226 | PORT.WRITE PORT.ADRS[CSR] WITH WR[0] | ;WRITE THE DATA TO THE CSR     |  |

```

:8227
:8228 READ.CSR.T5.4:
U 0873, 9090,15 :8229 MISC [SET.PORT.1.0] ;SELECT THE IDC
U 0874, 9780,15 :8230 PORT.READ PORT.ADRS[CSR] ;SET UP TO READ THE CSR
U 0875, 3B32,15 :8231 MOV PORT TO LS[PORT0] ;READ THE DATA FROM CSR AND
:8232 ; SAVE IT IN LS
U 0876, 3732,15 :8233 MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
U 0877, 377E,95 :8234 MOV LS[CSR.PAT.4] TO WR[1] ;SET UP EXPECTED DATA
U 0878, 0869,3C :8235 JSR [CHECK.RESULT] ;CHECK THE DATA
U 0879, 0887,14 :8236 JMP [LOOP.T5.4] ;LOOP HERE IF ERR & LOE IS SET
:8237
:8238 LOOP.T5.5:
U 087A, 3780,15 :8239 MOV LS[CSR.PAT.5] TO WR[0] ;GET THE DATA PATTERN
:8240
:8241 WRITE.CSR.T5.5:
U 087B, 17A0,15 :8242 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE DATA TO THE CSR
:8243
:8244 READ.CSR.T5.5:
U 087C, 9090,15 :8245 MISC [SET.PORT.1.0] ;SELECT THE IDC
U 087D, 9780,15 :8246 PORT.READ PORT.ADRS[CSR] ;SET UP TO READ THE CSR
U 087E, 3B32,15 :8247 MOV PORT TO LS[PORT0] ;READ THE DATA FROM CSR AND
:8248 ; SAVE IT IN LS
U 087F, 3732,15 :8249 MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
U 0880, B780,95 :8250 MOV LS[CSR.PAT.5] TO WR[1] ;SET UP EXPECTED DATA
U 0881, 0869,3C :8251 JSR [CHECK.RESULT] ;CHECK THE DATA
U 0882, 8887,A4 :8252 JMP [LOOP.T5.5] ;LOOP HERE IF ERR & LOE IS SET
:8253
:8254 CHECK.CLEAR22.T5:
U 0883, B784,15 :8255 MOV LS[BIT22.MASK] TO WR[0] ;GET MASK TO CHECK BIT 22
U 0884, 3E8A,15 :8256 MOV WR[0] TO LS[ERROR.MASK] ;SET IT
U 0885, FF82,15 :8257 INC LS[ERROR.NUMBER] ;INCREMENT TO ERROR 2
:8258
:8259 LOOP.T5.6:
U 0886, B712,15 :8260 MOV LS[SETCSR.CRDY] TO WR[0] ;CRDY MUST BE SET
U 0887, 476C,15 :8261 BIS LS[BIT22] TO WR[0] ;DATA TO WRITE IN WR 0
U 0888, 4778,15 :8262 BIS LS[BIT28] TO WR[0] ;SET THE TIMEOUT INHIBIT BIT
U 0889, 17A0,15 :8263 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE TO THE CSR
U 088A, 17CC,15 :8264 PORT.CONTROL [CLEAR.IDC] ;ISSUE INIT TO THE IDC
U 088B, 17CC,15 :8265 PORT.CONTROL [CLEAR.IDC]
U 088C, 17CC,15 :8266 PORT.CONTROL [CLEAR.IDC]
U 088D, 9090,15 :8267 MISC [SET.PORT.1.0] ;SELECT THE IDC
U 088E, 9780,15 :8268 PORT.READ PORT.ADRS[CSR] ;SET UP TO READ THE CSR
U 088F, 3B32,15 :8269 MOV PORT TO LS[PORT0] ;READ THE DATA FROM CSR AND
:8270 ; SAVE IT IN LS
U 0890, 3732,15 :8271 MOV LS[PORT0] TO WR[0] ;SET UP THE RECEIVED DATA
U 0891, B69C,95 :8272 MOV LS[ZERO] TO WR[1] ;SET THE EXPECTED DATA
U 0892, 0869,3C :8273 JSR [CHECK.RESULT] ;COMPARE THE DATA
U 0893, 8888,64 :8274 JMP [LOOP.T5.6] ;LOOP HERE IF ERR & LOE IS SET
:8275
:8276 CHECK.CLEAR23.T5:
U 0894, 3786,15 :8277 MOV LS[BIT23.MASK] TO WR[0] ;GET MASK TO CHECK BIT 23
U 0895, 3E8A,15 :8278 MOV WR[0] TO LS[ERROR.MASK] ;SET IT
U 0896, FF82,15 :8279 INC LS[ERROR.NUMBER] ;INCREMENT TO ERROR 3
:8280
:8281 LOOP.T5.7:

```

ZZ  
 :  
 :  
 U  
 U  
 U  
 U  
 U  
 U  
 U  
 U

|                 |       |                                      |                                |
|-----------------|-------|--------------------------------------|--------------------------------|
| U 0897, B712,15 | :8282 | MOV LS[SETCSR.CRDY] TO WR[0]         | ;CRDY MUST BE SET              |
| U 0898, C76E,15 | :8283 | BIS LS[BIT23] TO WR[0]               | ;DATA TO WRITE IN WR 0         |
| U 0899, 4778,15 | :8284 | BIS LS[BIT28] TO WR[0]               | ;SET THE TIMEOUT INHIBIT BIT   |
| U 089A, 17A0,15 | :8285 | PORT.WRITE PORT.ADRS[CSR] WITH WR[0] | ;WRITE TO THE CSR              |
| U 089B, 17CC,15 | :8286 | PORT.CONTROL [CLEAR.IDC]             | ;ISSUE INIT TO THE IDC         |
| U 089C, 17CC,15 | :8287 | PORT.CONTROL [CLEAR.IDC]             |                                |
| U 089D, 17CC,15 | :8288 | PORT.CONTROL [CLEAR.IDC]             |                                |
| U 089E, 9090,15 | :8289 | MISC [SET.PORT.I.O]                  | ;SELECT THE IDC                |
| U 089F, 9780,15 | :8290 | PORT.READ PORT.ADRS[CSR]             | ;SET UP TO READ THE CSR        |
| U 08A0, 3B32,15 | :8291 | MOV PORT TO LS[PORT0]                | ;READ THE DATA FROM CSR AND    |
|                 | :8292 |                                      | ; SAVE IT IN LS                |
| U 08A1, 3732,15 | :8293 | MOV LS[PORT0] TO WR[0]               | ;SET UP THE RECEIVED DATA      |
| U 08A2, 2F82,95 | :8294 | CLR WR[1]                            | ;SET THE EXPECTED DATA         |
| U 08A3, 0869,3C | :8295 | JSR [CHECK.RESULT]                   | ;COMPARE THE DATA              |
| U 08A4, 8889,74 | :8296 | JMP [LOOP.T5.7]                      | ;LOOP HERE IF ERR & LOE IS SET |
|                 | :8297 |                                      |                                |
| U 08A5, 3678,15 | :8298 | MOV LS[BIT28] TO WR[0]               |                                |
| U 08A6, B64E,95 | :8299 | MOV LS[BIT7] TO WR[1]                |                                |
| U 08A7, AEC2,15 | :8300 | BIS WR[1] TO WR[0]                   |                                |
| U 08A8, 17A0,15 | :8301 | PORT.WRITE PORT.ADRS[CSR] WITH WR[0] | ;SET WRITE INHIBIT             |
|                 | :8302 |                                      | ; AND CRDY                     |
|                 | :8303 |                                      |                                |
| U 08A9, 8875,3C | :8304 | JSR [CLEANUP]                        | ;CLEANUP THE CONDITIONS SET    |
|                 | :8305 |                                      | ; BY THIS TEST                 |
|                 | :8306 |                                      | ;END TEST                      |
|                 | :8307 |                                      |                                |
|                 | :8308 |                                      |                                |

END.T5:



U  
U  
U  
U  
U

```

8364 : 18. Check that CSR <CRDY> is set.
8365 : 19. End test.
8366 :
8367 : Errors:
8368 :
8369 : ERROR 1 - CSR <CRDY> did not clear.
8370 : Printout:
8371 : EXPECTED RECEIVED
8372 : CSR CSR
8373 :
8374 : ERROR 2 - DAR failed to increment correctly when an IDC microword
8375 : with INCR.DAR/ON was executed or the microcode did not
8376 : branch correctly.
8377 :
8378 : Printout:
8379 : EXPECTED RECEIVED
8380 : DAR DAR
8381 :
8382 : ERROR 3 - CSR <CRDY> did not set.
8383 : Printout:
8384 : EXPECTED RECEIVED
8385 : CSR CSR
8386 :
8387 : ERROR 4 - CSR <CRDY> did not clear.
8388 : Printout:
8389 : EXPECTED RECEIVED
8390 : CSR CSR
8391 :
8392 : ERROR 5 - CSR <CRDY> did not set when a CLEAR.IDC was issued.
8393 : Printout:
8394 : EXPECTED RECEIVED
8395 : CSR CSR
8396 :
8397 : Debug:
8398 :
8399 : ERROR 1:
8400 :
8401 : The PORT.WRITE PORT.ADRS[CSR] instruction will be decoded by the
8402 : PORT RD/WR DECODE PAL and will assert WRITE CSR L. This signal
8403 : and BUS IO 07 H are inputs to the CONTROL STATUS REG PAL. BUS IO 07 H
8404 : will be L and when P2 CLOCK L clocks, output CRDY L will be L.
8405 :
8406 : When the CSR is read, CRDY is fed through a driver that is enabled by
8407 : SEL CSR L which will be asserted on the decode of the READ CSR
8408 : instruction. This will output CRDY L onto BUS IO 07 H.
8409 :
8410 : ERROR 2:
8411 :
8412 : The CLEAR.IDC instruction will be decoded in the PORT RD/WR DECODE
8413 : PAL. The inputs will be:
8414 : CSR 17 H = H
8415 : CSR 14 H = H
8416 : CSR 12 H = L
8417 : CSR 11 H = H
8418 : CSR 10 H = H

```

```

:8419 : PORT INSTR H = H
:8420 :
:8421 : This combination of inputs will assert CLEAR IDC L. This signal is
:8422 : an input to the INIT rlip Flop. After 2 P2 CLOCKS, INIT H will
:8423 : assert. This signal is an input to the DAR and when asserted will
:8424 : reset the DAR = 0.
:8425 : When the INCR.DAR/ON microword is executed, UINCR DAR L will assert.
:8426 : This will increment the DAR.
:8427 :
:8428 : If the IDC microcode is not at DIAG.IDLE, then either the BEN0/CRDY L
:8429 : or the BEN2/MAINT H branch did not execute correctly.
:8430 :
:8431 : ERROR 3:
:8432 :
:8433 : The PORT.WRITE PORT.ADRS[CSR] instruction will be decoded by the
:8434 : PORT RD/WR DECODE PAL and will assert WRITE CSR L. This signal
:8435 : and BUS IO 07 H are inputs to the CONTROL STATUS REG PAL. BUS IO 07 H
:8436 : will be H and when P2 CLOCK L clocks, output CRDY L will be H.
:8437 :
:8438 : When the CSR is read, CRDY is fed through a driver that is enabled by
:8439 : SEL CSR L which will be asserted on the decode of the READ CSR
:8440 : instruction. This will output CRDY L onto BUS IO 07 H.
:8441 :
:8442 : ERROR 4:
:8443 :
:8444 : The PORT.WRITE PORT.ADRS[CSR] instruction will be decoded by the
:8445 : PORT RD/WR DECODE PAL and will assert WRITE CSR L. This signal
:8446 : and BUS IO 07 H are inputs to the CONTROL STATUS REG PAL. BUS IO 07 H
:8447 : will be L and when P2 CLOCK L clocks, output CRDY L will be L.
:8448 :
:8449 : When the CSR is read, CRDY is fed through a driver that is enabled by
:8450 : SEL CSR L which will be asserted on the decode of the READ CSR
:8451 : instruction. This will output CRDY L onto BUS IO 07 H.
:8452 :
:8453 : ERROR 5:
:8454 :
:8455 :
:8456 : The CLEAR.IDC instruction will be decoded in the PORT RD/WR DECODE
:8457 : PAL. The inputs will be:
:8458 : CSR 17 H = H
:8459 : CSR 14 H = H
:8460 : CSR 12 H = L
:8461 : CSR 11 H = H
:8462 : CSR 10 H = H
:8463 : PORT INSTR H = H
:8464 :
:8465 : This combination of inputs will assert CLEAR IDC L. This signal is
:8466 : an input to the INIT Flip Flop. After 2 P2 CLOCKS, INIT H will
:8467 : assert. INIT H is inverted and becomed INIT L. This signal is an input
:8468 : to the CONTROL STATUS REG PAL and when asserted will set CRDY L = H.
:8469 :
:8470 : When the CSR is read, CRDY is fed through a driver that is enabled by
:8471 : SEL CSR L which will be asserted on the decode of the READ CSR
:8472 : instruction. This will output CRDY L onto BUS IO 07 H.
:8473 :

```

B  
C  
C  
O  
D  
E  
D  
I  
N  
G  
I  
N  
F  
O  
R  
M  
A  
T  
I  
O  
N  
I  
S  
N  
O  
T  
T  
O  
B  
E  
R  
E  
P  
R  
O  
D  
U  
C  
E  
D  
O  
R  
R  
E  
C  
T  
E  
D  
I  
N  
T  
H  
I  
S  
D  
O  
C  
U  
M  
E  
N  
T

```

:8474 :--
:8475 :*****
:8476 T.6:
U 08AA, B65E,15 :8477 MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT 15 IN WR[0] FOR
:8478 : CONTROL AND STATUS WORD
U 08AB, 3E80,15 :8479 MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND
:8480 : STATUS WORD - BIT 15
:8481 : INDICATES START OF TEST TO
:8482 : CONSOLE
U 08AC, 10E0,15 :8483 MISC [SET.CP.ATTN] ;ISSUE CPU ATTN TO CONSOLE
:8484 WAIT.T6.0:
U 08AD, 888A,D4 :8485 JMP [WAIT.T6.0] ;LOOP HERE WAITING FOR
:8486 : CONSOLE TO RESPOND
U 08AE, 65A0,15 :8487 CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
:8488
U 08AF, B64A,15 :8489 MOV LS[IDC] TO WR[0] ;STORE MODULE CODE IN LS TO
U 08B0, 3E8C,15 :8490 MOV WR[0] TO LS[MODULE.NUM] ; INDICATE IDC MODULE
:8491
:8492 LOOP.T6.1:
U 08B1, B640,15 :8493 MOV LS[#1] TO WR[0] ;SET WRO = 1
U 08B2, BE82,15 :8494 MOV WR[0] TO LS[ERROR.NUMBER] ;SET UP ERROR NUMBER = 1 IN LS
:8495
U 08B3, B782,15 :8496 MOV LS[BIT7.MASK] TO WR[0] ;GET THE ERROR MASK TO CHECK
:8497 : BIT <7>
U 08B4, 3E8A,15 :8498 MOV WR[0] TO LS[ERROR.MASK] ;SET UP THE ERROR MASK TO CHECK
:8499 : TO CHECK CSR<CRDY>
:8500
U 08B5, 17CC,15 :8501 PORT.CONTROL [CLEAR.IDC] ;FORCE THE IDC TO JAM AND THEN
:8502 : ENTER IDLE
U 08B6, 17CC,15 :8503 PORT.CONTROL [CLEAR.IDC]
U 08B7, 17CC,15 :8504 PORT.CONTROL [CLEAR.IDC] ;DAR WILL BE CLEARED
:8505
U 08B8, 2F80,15 :8506 CLR WR[0]
U 08B9, 3720,15 :8507 MOV LS[SETCSR.MAINT] TO WR[0] ;GET DATA TO SET CSR MAINT BIT
:8508 : CRDY = 0 AND MAINT = 1
U 08BA, 4778,15 :8509 BIS LS[BIT28] TO WR[0] ;SET THE TIMEOUT INHIBIT BIT
U 08BB, 17A0,15 :8510 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE TO THE IDC CSR
:8511
U 08BC, B730,15 :8512 MOV LS[#1A] TO WR[0] ;SET UP A COUNTER TO NOP UNTIL
:8513 : IDC COMPLETES ITS OPERATION
U 08BD, 2100,15 :8514 NOP.T6: DEC WR[0]
:8515 :
:8516 CMP WR[0] WITH LS[ZERO],
:8517 DT(LONG)&SET.ALU.CC
:8518 JMP.IF[NEQ] TO [NOP.T6]
:8519
:8520 :-----
:8521 :*****
:8522 : * IDC *
:8523 :*****
:8524 :-----
:8525
:8526 : The IDC should be looping in the IDLE LOOP at this time. When the WRITE to
:8527 : IDC CSR is issued with CRDY clear and MAINT set, the IDC should branch
:8528 : out of the IDLE LOOP and then increment the DAR and then go to the

```

```

:8529 : diagnostic idle loop DIAG.IDLE and remain there because CSR F<2:0> = 0.
:8530 :-----
:8531 :=1
:8532 :TEST.OR.RUN:
:8533 :
:8534 : BEN2/MAINT,
:8535 : NAD/DRIVE.TYPE
:8536 :-----
:8537 :=1**
:8538 :UDIAGNOSTICS:
:8539 :
:8540 : INCR.DAR/ON, ;INCREMENT THE DAR AND THEN JUMP TO
:8541 : NAD/DIAG.IDLE ; THE DIAGNOSTIC IDLE LOOP
:8542 :-----
:8543 :018:
:8544 :=000
:8545 :DIAG.IDLE:
:8546 :
:8547 : BEN0/F0,
:8548 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:8549 : BEN2/F2,
:8550 : NAD/DIAG.IDLE
:8551 :-----
:8552 :-----
:8553 :
U 08C0, 9090,15 :8554 MISC [SET.PORT.I.0] ;SELECT THE PORT TO BUS
U 08C1, 9780,15 :8555 PORT.READ PORT.ADRS[CSR] ;READ THE CSR
U 08C2, 3B32,15 :8556 MOV PORT TO LS[PORT0] ;MOV THE DATA TO LS
U 08C3, 3732,15 :8557 MOV LS[PORT0] TO WR[0] ;SAVE THE RECEIVED DATA
U 08C4, 2F82,95 :8558 CLR WR[1] ;GET THE EXPECTED DATA
U 08C5, 0869,3C :8559 JSR [CHECK.RESULT] ;COMPARE
U 08C6, 088B,14 :8560 JMP [LOOP.T6.1] ;LOOP HERE IF ERR & LOE IS SET
U 08C7, 3642,15 :8561 MOV LS[#2] TO WR[0] ;SET UP FOR ERROR 2
U 08C8, BE82,15 :8562 MOV WR[0] TO LS[ERROR.NUMBER]
:8563
U 08C9, 3776,15 :8564 MOV LS[DAR.MASK] TO WR[0] ;GET THE MASK TO CHECK BITS
U 08CA, 3E8A,15 :8565 MOV WR[0] TO LS[ERROR.MASK] ; <18:0> OF THE DAR
:8566
U 08CB, 9090,15 :8567 MISC [SET.PORT.I.0] ;SELECT PORT TO BUS
U 08CC, 1784,15 :8568 PORT.READ PORT.ADRS[DAR] ;READ THE DISK ADDRS REG
U 08CD, 3B32,15 :8569 MOV PORT TO LS[PORT0] ;PUT THE REC DATA IN LS
U 08CE, 3732,15 :8570 MOV LS[PORT0] TO WR[0] ;SET UP TO PASS REC DATA
U 08CF, 3640,95 :8571 MOV LS[#1] TO WR[1] ;SET EXP DATA
U 08D0, 0869,3C :8572 JSR [CHECK.RESULT] ;GO CHECK THE DATA
U 08D1, 088B,14 :8573 JMP [LOOP.T6.1] ;LOOP HERE IF ERR & LOE IS SET
:8574
:8575 CHECK.CRDYSET.T6:
U 08D2, FF82,15 :8576 INC LS[ERROR.NUMBER]
U 08D3, B782,15 :8577 MOV LS[BIT7.MASK] TO WR[0] ;GET THE MASK TO CHECK ONLY
U 08D4, 3E8A,15 :8578 MOV WR[0] TO LS[ERROR.MASK] ; BIT 7
:8579
:8580 LOOP.T6.2:
U 08D5, 364E,15 :8581 MOV LS[BIT7] TO WR[0] ;BIT 7 <CRDY> SET
U 08D6, 4778,15 :8582 BIS LS[BIT28] TO WR[0] ;SET THE TIMEOUT INHIBIT BIT
U 08D7, 17A0,15 :8583 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;SET CSR <CRDY>

```

C 16

ZZ-ENKCF-1.4 ; ENKCF.MIC Test 6 CRDY & IDC 15-MAR-83 Fiche 1 Frame C16 Sequence 197  
: ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40 Page 191  
: ENKCF.MIC Test 6 CRDY & IDC DIAGNOSTIC MICROCODE ACCESS TEST (M8388 IDC MODULE)

```

U 08D8, 9090,15 :8584 MISC [SET.PORT.1.0] ;SELECT THE PORT TO BUS
U 08D9, 9780,15 :8585 PORT.READ PORT.ADRS[CSR] ;READ THE CSR
U 08DA, 3B32,15 :8586 MOV PORT TO LS[PORT0] ;MOV THE DATA TO LS
U 08DB, 3732,15 :8587 MOV LS[PORT0] TO WR[0] ;SAVE THE RECEIVED DATA
U 08DC, B64E,95 :8588 MOV LS[BIT7] TO WR[1] ;SET BIT 7 IN EXPECTED DATA
U 08DD, 0869,3C :8589 JSR [CHECK.RESULT] ;COMPARE THE DATA
U 08DE, 888D,54 :8590 JMP [LOOP.T6.2] ;LOOP HERE IF ERR & LOE IS SET
:8591
:8592 CHECK.CRDYCLR.T6:
U 08DF, FF82,15 :8593 INC LS[ERROR.NUMBER] ;SET UP FOR ERROR 4
U 08E0, B782,15 :8594 MOV LS[BIT7.MASK] TO WR[0] ;GET THE MASK TO CHECK ONLY
U 08E1, 3E8A,15 :8595 MOV WR[0] TO LS[ERROR.MASK] ; BIT 7
:8596
:8597 LOOP.T6.3:
U 08E2, 2F80,15 :8598 CLR WR[0] ;CLR BIT 7
U 08E3, 4778,15 :8599 BIS LS[BIT28] TO WR[0] ;SET THE TIMEOUT INHIBIT BIT
U 08E4, 17A0,15 :8600 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;CLEAR CSR <CRDY>
U 08E5, 9090,15 :8601 MISC [SET.PORT.1.0] ;SELECT THE PORT TO BUS
U 08E6, 9780,15 :8602 PORT.READ PORT.ADRS[CSR] ;READ THE CSR
U 08E7, 3B32,15 :8603 MOV PORT TO LS[PORT0] ;MOV THE DATA TO LS
U 08E8, 3732,15 :8604 MOV LS[PORT0] TO WR[0] ;SAVE THE RECEIVED DATA
U 08E9, 2F82,95 :8605 CLR WR[1] ;GET THE EXPECTED DATA
U 08EA, 0869,3C :8606 JSR [CHECK.RESULT] ;COMPARE THE DATA
U 08EB, 088E,24 :8607 JMP [LOOP.T6.3] ;LOOP HERE IF ERR & LOE IS SET
:8608
:8609 CHECK.CLEARIDC.T6:
U 08EC, FF82,15 :8610 INC LS[ERROR.NUMBER] ;SET UP FOR ERROR 5
U 08ED, B782,15 :8611 MOV LS[BIT7.MASK] TO WR[0] ;GET THE MASK TO CHECK ONLY
U 08EE, 3E8A,15 :8612 MOV WR[0] TO LS[ERROR.MASK] ; BIT 7
:8613
:8614 LOOP.T6.4:
U 08EF, 17CC,15 :8615 PORT.CONTROL [CLEAR.IDC] ;INITIALIZE THE IDC
U 08F0, 17CC,15 :8616 PORT.CONTROL [CLEAR.IDC]
U 08F1, 17CC,15 :8617 PORT.CONTROL [CLEAR.IDC]
U 08F2, 9090,15 :8618 MISC [SET.PORT.1.0] ;SELECT THE PORT TO BUS
U 08F3, 9780,15 :8619 PORT.READ PORT.ADRS[CSR] ;READ THE CSR
U 08F4, 3B32,15 :8620 MOV PORT TO LS[PORT0] ;MOV THE DATA TO LS
U 08F5, 3732,15 :8621 MOV LS[PORT0] TO WR[0] ;SAVE THE RECEIVED DATA
U 08F6, B64E,95 :8622 MOV LS[BIT7] TO WR[1] ;GET THE EXPECTED DATA
U 08F7, 0869,3C :8623 JSR [CHECK.RESULT] ;COMPARE THE DATA
U 08F8, 888E,F4 :8624 JMP [LOOP.T6.4] ;LOOP HERE IF ERR & LOE IS SET
:8625
U 08F9, 8875,3C :8626 JSR [CLEANUP] ;CLEANUP CONDITIONS SET BY THIS
:8627 ; TEST
:8628
:8629 END.T6:
:8630
:8631
:8632
:8633

```

```

:8634 :PAGE "Test 7 CSR BITS <F0><F1><F2> BRANCH TEST (M8388 IDC MODULE)"
:8635 :++
:8636 :
:8637 : Test Description:
:8638 :
:8639 : This test will verify BEN0/F0 H, BEN1/F1 H and BEN2/F2 H branch
:8640 : conditions. The INCR.DAR function will be used to verify that the
:8641 : microcode branches correctly.
:8642 :
:8643 : The data patterns that will be used are:
:8644 : CSR F<2:0> = 001
:8645 : CSR F<2:0> = 010
:8646 : CSR F<2:0> = 100
:8647 :
:8648 :
:8649 : Assumptions:
:8650 :
:8651 : It is assumed that the INCR.DAR function works correctly.
:8652 : It is also assumed that the IDC microcode is looping in DIAG.IDLE.
:8653 : DIAG.IDLE is defined as follows:
:8654 : BEN0/F0 H
:8655 : BEN1/F1 H
:8656 : BEN2/F2 H
:8657 :
:8658 : Test Steps:
:8659 :
:8660 : 1. Set up error mask, error number and module number in LS
:8661 : (for error printouts) and clear the previous error number in LS.
:8662 : 2. Force the IDC microcode to DIAG.IDLE.
:8663 : 3. Clear the DAR.
:8664 : 4. Load the data pattern into CSR F<2:0>.
:8665 : 5. Set up CSR bits <CRDY> and <MAINT> to branch to the correct
:8666 : microcode routine.
:8667 :
:8668 :
:8669 : *****
:8670 : * IDC *
:8671 : *****
:8672 :
:8673 : IDC1. Execute an instruction with INCR.DAR/ON.
:8674 : IDC2. Return to DIAG.IDLE after the CPU has cleared CSR F<2:0>.
:8675 :
:8676 : 6. NOP to allow IDC microcode to execute.
:8677 : 7. Clear CSR F<2:0>.
:8678 : 8. Check the DAR = 1.
:8679 : 9. Increment the error number.
:8680 : 10. Get the next data pattern and go to step 2 until all data patterns
:8681 : have been tested.
:8682 : 11. Clear IDC.
:8683 : 12. End test.
:8684 :
:8685 : Errors:
:8686 :
:8687 : Printout:
:8688 : EXPECTED RECEIVED

```

```

:8689 : DAR DAR
:8690 :
:8691 : ERROR 1 - The microcode did not branch correctly with CSR F<2:0> = 001.
:8692 :
:8693 : ERROR 2 - The microcode did not branch correctly with CSR F<2:0> = 010.
:8694 :
:8695 : ERROR 3 - The microcode did not branch correctly with CSR F<2:0> = 100.
:8696 :
:8697 :
:8698 : Debug:
:8699 :
:8700 : The IDC microcode will be looping at DIAG.IDLE which is defined as
:8701 : BEN0/F0 H, BEN1/F1 H and BEN2/F2 H. CSR F<2:0> will be clear so the
:8702 : microcode will remain at DIAG.IDLE until any of the function bits are
:8703 : changed.
:8704 :
:8705 : ERROR 1:
:8706 :
:8707 : The CSR is written with the data pattern, F<2:0>=001.
:8708 : BEN0/F0 H will be selected. F0 H will be asserted which will cause the
:8709 : microcode to branch to a word which contains INCR.DAR/ON. This will
:8710 : assert UINCR DAR L which will increment the DAR.
:8711 : The IDC microcode will then jump to DIAG.IDLE.
:8712 :
:8713 : ERROR 2:
:8714 :
:8715 : Same as ERROR 1 except the data pattern is CSR F<2:0>= 010.
:8716 : BEN1/F1 H will be selected and F1 H will be asserted.
:8717 :
:8718 : ERROR 3:
:8719 :
:8720 : Same as ERROR 1 except the data pattern is CSR F<2:0>= 100.
:8721 : BEN2/F2 H will be selected and F2 H will be asserted.
:8722 :
:8723 : --
:8724 : *****
:8725 : T.7: MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT 15 IN WR[0] FOR
:8726 : ; CONTROL AND STATUS WORD
:8727 : MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND
:8728 : ; STATUS WORD - BIT 15
:8729 : ; INDICATES START OF TEST TO
:8730 : ; CONSOLE
:8731 : MISC [SET.CP.ATTN] ;ISSUE CPU ATTN TO CONSOLE
:8732 : WAIT.T7.0:
:8733 : JMP [WAIT.T7.0] ;LOOP HERE WAITING FOR
:8734 : ; CONSOLE TO RESPOND
:8735 :
:8736 : CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
:8737 : ; IN LS
:8738 :
:8739 : MOV LS[IDC] TO WR[0] ;STORE MODULE CODE IN LS TO
:8740 : MOV WR[0] TO LS[MODULE.NUM] ; INDICATE IDC MODULE
:8741 :
:8742 :
:8743 : MOV LS[#1] TO WR[0] ;SET WRO = 1

```

```

U 08FA, B65E,15
U 08FB, 3E80,15
U 08FC, 10E0,15
U 08FD, 888F,D4
U 08FE, 65A0,15
U 08FF, B64A,15
U 0900, 3E8C,15
U 0901, B640,15

```



```

U 0902, BE82,15 :8744 MOV WR[0] TO LS[ERROR.NUMBER] ;SET UP ERROR NUMBER = 1 IN LS
 :8745
U 0903, 3776,15 :8746 MOV LS[DAR.MASK] TO WR[0] ;GET THE MASK TO CHECK
U 0904, 3E8A,15 :8747 MOV WR[0] TO LS[ERROR.MASK] ; BITS <18:0> OF THE DAR
 :8748
 :8749
 :8750
U 0905, 8872,EC :8751 LOOP.T7.1:
 :8752 JSR [DIAG.CODE] ;FORCE THE IDC MICROCODE TO
 :8753 ; THE DIAGNOSTIC IDLE LOOP
 :8754 ; (DIAG.IDLE)
 :8755
U 0906, 2F80,15 :8756 CLR WR[0] ;SET UP DATA TO CLR DAR
U 0907, 97A4,15 :8757 PORT.WRITE PORT.ADRS[DAR] WITH WR[0] ;CLR THE DAR
 :8758
U 0908, 3720,15 :8759 MOV LS[SETCSR.MAINT] TO WR[0] ;SET UP THE CSR TO BRANCH TO
 :8760 ; THE CORRECT IDC MICRO ROUTINE
U 0909, B702,95 :8761 MOV LS[SETCSR.F1] TO WR[1] ;SET FUNC BITS = 1 TO TEST
U 090A, AEC2,15 :8762 BIS WR[1] TO WR[0] ;PUT DATA IN WR 0 FOR WRITE
U 090B, 17A0,15 :8763 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
 :8764 ;MAINT=1,CRDY=0,F<2:0>=001
 :8765
 :8766
 :8767
 :8768 *****
 :8769 * IDC *
 :8770 *****
 :8771
 :8772 :018:
 :8773 =000
 :8774 :DIAG.IDLE:
 :8775
 :8776 BEN0/F0,
 :8777 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
 :8778 BEN2/F2,
 :8779 NAD/DIAG.IDLE
 :8780
 :8781 :019:
 :8782 =001
 :8783
 :8784 BEN0/CRDY.L,
 :8785 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
 :8786 NAD/TEST.FUNC1 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
 :8787
 :8788 :065:
 :8789 =1*1
 :8790 :TEST.F1:
 :8791 INCR.DAR/ON, ;INCREMENT THE DAR
 :8792 NAD/XFER.REQ ;SEND XFER.REQ
 :8793
 :8794
 :8795
U 090C, B700,15 :8796 MOV LS[SETCSR.F0] TO WR[0] ;SET UP THE CLEAR CSR FUNC BITS
U 090D, 17A0,15 :8797 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
U 090E, DB00,15 :8798 NOP ;WAIT FOR IDC TO COMPLETE

```

```

:8799
U 090F, 9090,15 :8800 MISC [SET.PORT.I.0] ;SELECT THE PORT TO BUS
U 0910, 1784,15 :8801 PORT.READ PORT.ADRS[DAR] ;READ THE DAR
U 0911, 3832,15 :8802 MOV PORT TO LS[PORT0] ;BRING IN THE DATA
U 0912, 3732,15 :8803 MOV LS[PORT0] TO WR[0] ;SAVE THE RECEIVED DATA
U 0913, 3640,95 :8804 MOV LS[#1] TO WR[1] ;GET THE EXPECTED DATA
U 0914, 0869,3C :8805 JSR [CHECK.RESULT] ;COMPARE THE DATA
U 0915, 8890,54 :8806 JMP [LOOP.T7.1] ;LOP HERE IF ERR & LOE IS SET
:8807
U 0916, FF82,15 :8808 INC LS[ERROR.NUMBER] ;SET UP FOR ERROR 2
:8809
:8810 LOOP.T7.2:
U 0917, 8872,EC :8811 JSR [DIAG.CODE] ;FORCE THE IDC MICROCODE TO
:8812 ; THE DIAGNOSTIC IDLE LOOP
:8813 ; (DIAG.IDLE)
:8814
U 0918, 2F80,15 :8815 CLR WR[0] ;SET UP DATA TO CLR DAR
U 0919, 97A4,15 :8816 PORT.WRITE PORT.ADRS[DAR] WITH WR[0] ;CLR THE DAR
:8817
U 091A, 3720,15 :8818 MOV LS[SETCSR.MAINT] TO WR[0] ;SET UP THE CSR TO BRANCH TO
:8819 ; THE CORRECT IDC MICRO ROUTINE
U 091B, 8704,95 :8820 MOV LS[SETCSR.F2] TO WR[1] ;SET FUNC BITS = 2 TO TEST
U 091C, AEC2,15 :8821 BIS WR[1] TO WR[0] ;SET BITS IN WR[0] FOR WRITE
U 091D, 17A0,15 :8822 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:8823 ;MAINT=1,CRDY=0,F<2:0>=010
:8824 -----
:8825 -----
:8826 ;*****
:8827 ; * IDC *
:8828 ;*****
:8829 -----
:8830 ;018:
:8831 ;=000
:8832 ;DIAG.IDLE:
:8833 ;
:8834 ;BEN0/F0,
:8835 ;BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:8836 ;BEN2/F2,
:8837 ;NAD/DIAG.IDLE
:8838 -----
:8839 ;01A:
:8840 ;=010
:8841 ;
:8842 ;BEN0/CRDY.L,
:8843 ;BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:8844 ;NAD/TEST.FUNC2 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:8845 -----
:8846 ;067:
:8847 ;=1*1
:8848 ;
:8849 ;TEST.F2:
:8850 ;INCR.DAR/ON, ;INCREMENT THE DAR
:8851 ;NAD/DIAG.IDLE ;AND RETURN TO DIAG.IDLE
:8852 -----
:8853

```

H 16  
Fiche 1 Frame H16  
Sequence 202  
Page 196

ZZ-ENKCF-1.4 : ENKCF.MIC Test 7 CSR BITS <F15>-MAR-83 ENKCF Micro-diagnostic for IDC module Rev 01.40  
: ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22  
: ENKCF.MIC Test 7 CSR BITS <F0><F1><F2> BRANCH TEST (M8388 IDC MODULE)

```

U 091E, B700,15 :8854 MOV LS[SETCSR.F0] TO WR[0] ;SET UP THE CLEAR CSR FUNC BITS
U 091F, 17A0,15 :8855 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
U 0920, D~70,15 :8856 NOP ;WAIT FOR IDC TO COMPLETE
:8857
U 0921, 9090,15 :8858 MISC [SET.PORT.I.0] ;SELECT THE PORT TO BUS
U 0922, 1784,15 :8859 PORT.READ PORT.ADRS[DAR] ;READ THE DAR
U 0923, 3B32,15 :8860 MOV PORT TO LS[PORT0] ;BRING IN THE DATA
U 0924, 3732,15 :8861 MOV LS[PORT0] TO WR[0] ;SAVE THE RECEIVED DATA
U 0925, 3640,95 :8862 MOV LS[#1] TO WR[1] ;GET THE EXPECTED DATA
:8863
U 0926, 0869,3C :8864 JSR [CHECK.RESULT] ;CHECK THE DATA
U 0927, 8891,74 :8865 JMP [LOOP.T7.2] ;LOOP HERE IF ERR & LOE IS SET
:8866
U 0928, FF82,15 :8867 INC LS[ERROR.NUMBER] ;SET UP FOR ERROR 3
:8868 LOOP.T7.3:
U 0929, 8872,EC :8869 JSR [DIAG.CODE] ;FORCE THE IDC MICROCODE TO
:8870 ; THE DIAGNOSTIC IDLE LOOP
:8871 ; (DIAG.IDLE)
:8872
U 092A, 2F80,15 :8873 CLR WR[0] ;SET UP DATA TO CLR DAR
U 092B, 97A4,15 :8874 PORT.WRITE PORT.ADRS[DAR] WITH WR[0] ;CLR THE DAR
:8875
U 092C, 3720,15 :8876 MOV LS[SETCSR.MAINT] TO WR[0] ;SET UP THE CSR TO BRANCH TO
:8877 ; THE CORRECT IDC MICRO ROUTINE
U 092D, B708,95 :8878 MOV LS[SETCSR.F4] TO WR[1] ;SET FUNC BITS = 4 TO TEST
U 092E, AEC2,15 :8879 BIS WR[1] TO WR[0] ;SET BITS IN WR[0] FOR WRITE
U 092F, 17A0,15 :8880 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:8881 ;MAINT=1,CRDY=0,F<2:0>=100
:8882
:8883 -----
:8884 ;*****
:8885 ; * IDC *
:8886 ;*****
:8887 -----
:8888 :018:
:8889 :=000
:8890 :DIAG.IDLE:
:8891
:8892 : BEN0/F0,
:8893 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:8894 : BEN2/F2,
:8895 : NAD/DIAG.IDLE
:8896 -----
:8897 :01C:
:8898 :=100
:8899
:8900 : BEN0/CRDY.L,
:8901 : BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:8902 : NAD/TEST.FUNC4 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:8903 -----
:8904 :06F:
:8905 :=1*1
:8906 :TEST.F4:
:8907 : UCMD/CHG.FIFO, ;CHANGE FIFO TO USEL FIFO B
:8908 : INCR.DAR/ON, ;INCREMENT THE DAR

```

```

:8909 : NAD/XFER.REQ ;SEND XFER REQ
:8910 :-----
:8911
:8912
U 0930, 8700,15 :8913 MOV LS[SETCSR.F0] TO WR[0] ;SET UP THE CLR THE FUNC BITS
U 0931, 17A0,15 :8914 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:8915
U 0932, 9090,15 :8916 MISC [SET.PORT.I.0] ;SELECT THE PORT TO BUS
U 0933, 1784,15 :8917 PORT.READ PORT.ADRS[DAR] ;READ THE DAR
U 0934, 3832,15 :8918 MOV PORT TO LS[PORT0] ;BRING IN THE DATA
U 0935, 3732,15 :8919 MOV LS[PORT0] TO WR[0] ;SAVE THE RECEIVED DATA
U 0936, 3640,95 :8920 MOV LS[#1] TO WR[1] ;GET THE EXPECTED DATA
:8921
U 0937, 0869,3C :8922 JSR [CHECK.RESULT] ;CHECK THE DATA
U 0938, 0892,94 :8923 JMP [LOOP.T7.3] ;LOOP HERE IF ERR & LOE IS SET
:8924
U 0939, 8875,3C :8925 JSR [CLEANUP] ;GO CLEANUP
:8926 END.T7: ;END TEST
:8927
:8928
:8929
:8930

```

:8931 :PAGE "Test 8 XFER.REQ-XFER GRANT TEST(M8388 IDC MODULE OR M8390 CPU MODULE)"  
 :8932 :++

:8933 :  
 :8934 : Test Description:

:8935 :  
 :8936 : This test will verify that the XFER REQUEST - XFER GRANT signals  
 :8937 : between the IDC and the CPU operate correctly. The CPU will first set  
 :8938 : up the CSR to cause the IDC microcode to branch to an instruction  
 :8939 : that will issue an XFER.REQ. The CPU will then NOP to allow the IDC  
 :8940 : time to set XFER.REQ. The CPU will then execute a  
 :8941 : SKIP.IF[NO.FAST.INTERRUPT] instruction to check that XFER.REQ is  
 :8942 : asserted.  
 :8943 : The CPU then will issue XFER GRANT which should clear the XFER.REQ.  
 :8944 : The CPU will execute a SKIP.IF[NO.FAST.INTERRUPT] instruction  
 :8945 : to check that the XFER.REQ was cleared.

:8946 :  
 :8947 : Assumptions:

:8948 :  
 :8949 : It is assumed that the CSR F<2:0> BRANCH TEST has run successfully.

:8950 :  
 :8951 : Test Steps:

- :8952 :  
 :8953 : 1. Set up error mask, error number and module number in LS  
 :8954 : (for error printouts) and clear the previous error number in LS.  
 :8955 : 2. Force the IDC microcode to DIAG.IDLE.  
 :8956 : 3. Set the CSR to branch to the correct IDC microcode routine.

:8957 :  
 :8958 : \*\*\*\*\*  
 :8959 : \* IDC \*  
 :8960 : \*\*\*\*\*

:8961 :  
 :8962 : IDC1. Execute a micro-instruction with UCM = 4 (SET.XFER.REQ)  
 :8963 : and go to DIAG.WAIT.

- :8964 :  
 :8965 : 4. NOP to allow time for IDC to set XFER.REQ.  
 :8966 : 5. Clear CSR F<2:0>.  
 :8967 : 6. Execute a SKIP.IF[NO.FAST.INTERRUPT] and check that XFER.REQ  
 :8968 : is asserted.  
 :8969 : 7. Increment the error number.  
 :8970 : 8. Send XFER.GRANT.  
 :8971 : 9. NOP to allow time for XFER.REQ to clear.  
 :8972 : 10. Execute a SKIP.IF[NO.FAST.INTERRUPT] and check that XFER.REQ  
 :8973 : was cleared.  
 :8974 : 11. Clear IDC.  
 :8975 : 12. End test.

:8976 :  
 :8977 : Errors:

:8978 :  
 :8979 :  
 :8980 : Printout:  
 :8981 : EXPECTED RECEIVED  
 :8982 : N/A N/A

:8983 :  
 :8984 : ERROR 1 - The IDC micro-instruction sequence did not properly execute  
 :8985 : to cause XFER REQ to set.

```

:8986 :
:8987 : ERROR 2 - A MISC2 [TRANSFER.GRANT] did not clear the XFER REQ.
:8988 :
:8989 : Debug:
:8990 :
:8991 : ERROR 1:
:8992 :
:8993 : At test step IDC1 time, UCM <20:18> = 0100. These bits are then decoded
:8994 : in the CONTROL STATUS REG PAL. The inputs to the CONTROL STATUS
:8995 : REG PAL will be:
:8996 : UCMD 2 H = H
:8997 : UCMD 1 H = L
:8998 : UCMD 0 H = L
:8999 : INIT L = H
:9000 : XFER GRANT L = H
:9001 : When the CONTROL STATUS REG PAL is clocked with P2 CLOCK,
:9002 : XFER REQ H will assert.
:9003 : This signal is then driven off the IDC module as PORT XFER REQ.
:9004 :
:9005 : PORT XFER REQ L is gated through a Flip-Flop on the CPU module and
:9006 : becomes PORT INT L. This signal is an input to the MUX & INCR CTL PAL
:9007 : on the CPU module. When decoded, the CPU should not SKIP when
:9008 : PORT INT L is asserted.
:9009 :
:9010 : ERROR 2 -
:9011 : The CPU will then clear CSR F<2:0> to cause the IDC microcode to remain
:9012 : at DIAG.IDLE when the GRANT is issued. A MISC2 [TRANSFER.GRANT]
:9013 : instruction is then executed. This instruction will be decoded
:9014 : on the CPU module by the MISC INST DECODER and will assert
:9015 : XFER GRANT L.
:9016 :
:9017 : XFER GRANT is an input to the CONTROL STATUS REG PAL.
:9018 : At the next P2 CLOCK, the output XFER REQ H will deassert which
:9019 : will cause PORT INT L (an input to the MUX & INCR CTL PAL) on the CPU
:9020 : module to deassert.
:9021 : The CPU will then execute a SKIP.IF[NO.FAST.INTERRUPT] instruction
:9022 : and the instruction will SKIP.
:9023 : --
:9024 : *****
:9025 : T.8:
:9026 : MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT 15 IN WR[0] FOR
:9027 : MOV WR[0] TO LS[CONTROL.STATUS] ;CONTROL AND STATUS WORD
:9028 : ;SET BIT 15 IN CONTROL AND
:9029 : ;STATUS WORD - BIT 15
:9030 : ;INDICATES START OF TEST TO
:9031 : ;CONSOLE
:9032 : MISC [SET.CP.ATTN] ;ISSUE CPU ATTN TO CONSOLE
:9033 : WAIT.T8.0:
:9034 : JMP [WAIT.T8.0] ;LOOP HERE WAITING FOR
:9035 : ;CONSOLE TO RESPOND
:9036 :
:9037 : CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
:9038 : ;IN LS
:9039 :
:9040 : MOV LS[CPU] TO WR[0] ;STORE MODULE CODE IN LS TO

```

```

U 093A, B65E,15
U 093B, 3E80,15
U 093C, 10E0,15
U 093D, 0893,D4
U 093E, 65A0,15
U 093F, 3642,15

```

```

 L 16
ZZ-ENKCF-1.4 : ENKCF.MIC Test 8 XFER.REQ-XF15-MAR-83 Fiche 1 Frame L16 Sequence 206
: ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40 Page 200
: ENKCF.MIC Test 8 XFER.REQ-XFER GRANT TEST(M8388 IDC MODULE OR M8390 CPU MODULE)

U 0940, C74A,15 :9041 BIS LS[IDC] TO WR[0] ; INDICATE IDC AND CPU
U 0941, 3E8C,15 :9042 MOV WR[0] TO LS[MODULE.NUM] ; MODULES
:9043
:9044
U 0942, B640,15 :9045 MOV LS[#1] TO WR[0] ;SET WRO = 1
U 0943, BE82,15 :9046 MOV WR[0] TO LS[ERROR.NUMBER] ;SET UP ERROR NUMBER = 1 IN LS
:9047
U 0944, E58A,15 :9048 CLR LS[ERROR.MASK]
U 0945, B66E,15 :9049 MOV LS[NA.EXP.REC] TO WR[0] ;SET UP TO PRINT N/A FOR
U 0946, 3E80,15 :9050 MOV WR[0] TO LS[CONTROL.STATUS] ; EXPECTED & RECEIVED DATA
:9051
U 0947, 8872,EC :9052 JSR [DIAG.CODE] ;FORCE THE IDC MICROCODE TO
:9053 ; DIAG.IDLE
:9054
LOOP.T8.1:
U 0948, B720,95 :9055 MOV LS[SETCSR.MAINT] TO WR[1] ;SET UP THE CSR TO BRANCH
U 0949, B706,15 :9056 MOV LS[SETCSR.F3] TO WR[0] ; TO THE CORRECT IDC ROUTINE
U 094A, AEC2,15 :9057 BIS WR[1] TO WR[0]
U 094B, 17A0,15 :9058 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:9059 ;MAINT=1,CRDY=0,F<2:0>=011
:9060
:9061
:9062 *****
:9063 * IDC *
:9064 *****
:9065
:9066 :018:
:9067 :=000
:9068 :DIAG.IDLE:
:9069
:9070 BEN0/F0,
:9071 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:9072 BEN2/F2,
:9073 NAD/DIAG.IDLE
:9074
:9075 :01B:
:9076 :=011
:9077 BEN0/CRDY.L,
:9078 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:9079 NAD/TEST.FUNC3 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:9080
:9081 :06D:
:9082 :=1*1
:9083 :TEST.F3:
:9084
:9085 ;MUST NOT INCR DAR BETWEEN HERE AND
:9086 ; WHEN RETURNS TO DIAG.WAIT
:9087 NAD/XFER.REQ ;GO SET XFER REQ REQ-GRANT TEST
:9088
:9089
U 094C, DB00,15 :9090 NOP ;WAIT FOR THE IDC
U 094D, DB00,15 :9091 NOP
U 094E, DB00,15 :9092 NOP
U 094F, DB00,15 :9093 NOP
U 0950, DB00,15 :9094 NOP
U 0951, DB00,1A :9095 SKIP.IF[NO.FAST.INTERRUPT] ;CHECK TO SEE IF XFER REQ

```

U  
U



9139 .PAGE "Test 9 UINCREMENT DAR TEST (M8388 IDC MODULE)"

9140 .\*\*\*

9141 .  
9142 . Test Description:

9143 .  
9144 . This test is designed to insure that the DAR increments correctly  
9145 . when a UINCR.DAR function is executed. The register will first be  
9146 . loaded with zero and checked. Then the DAR will be incremented once  
9147 . and the contents checked until the entire register has been tested.

9148 .  
9149 . Assumptions:

9150 .  
9151 . It is assumed that the DAR REGISTER TEST, the XFER.REQ - XFER GRANT  
9152 . TEST and the CSR F<2:0> BRANCH TEST have run successfully.

9153 .  
9154 . Test Steps:

- 9155 .  
9156 . 1. Set up error mask, error number and module number in LS  
9157 . (for error printouts) and clear the previous error number in LS.  
9158 . 2. Force the IDC microcode to DIAG.IDLE.  
9159 . 3. Clear LS[T8]. Used to hold the expected data.  
9160 . 4. Load the DAR with zero.  
9161 . 5. Check the DAR = 0.  
9162 . 6. Increment the error number.  
9163 . 7. Set up the CSR the branch to the correct IDC microcode routine.

9164 .  
9165 .  
9166 . \*\*\*\*\*  
9167 . \* IDC \*  
9168 . \*\*\*\*\*

9169 .  
9170 . IDC1. Execute an instruction with INCR.DAR/ON and issue  
9171 . XFER.REQ to flag the CPU.

- 9172 .  
9173 . 8. Wait for a XFER.REQ from IDC.  
9174 . 9. Clear CSR F<2:0> and send XFER GRANT to clear the XFER.REQ.  
9175 . 10. Check the contents of the DAR.  
9176 . 11. Repeat steps 7 - 11 until the entire register has been checked.  
9177 . 12. Clear IDC.  
9178 . 13. End test.

9179 .  
9180 . Errors:

9181 .  
9182 .  
9183 . Printout:  
9184 . EXPECTED RECEIVED  
9185 . DAR DAR  
9186 .  
9187 . ERROR 1 - The DAR did not clear.  
9188 .  
9189 . ERROR 2 - The DAR did not increment correctly.

9190 .  
9191 . Debug:

9192 .  
9193 . ERROR 1 -

```

:9194 : WRITE.DAR:
:9195 : To allow the data pattern to be written into the DAR, the inputs to
:9196 : the PORT RD/WR DECODE PAL will be:
:9197 : CSR 17 H = H
:9198 : CSR 14 H = L
:9199 : CSR 13 H = H
:9200 : CSR 12 H = L
:9201 : CSR 11 H = L
:9202 : CSR 10 H = H
:9203 : PORT INSTR H = H
:9204 : These inputs will assert WRITE DAR L = L, which will load the
:9205 : DAR on the next Current Clk pulse with the data which is on the I/O
:9206 : BUS.
:9207 :
:9208 : READ.DAR:
:9209 : To read the DAR, the PORT INTERFACE REG PAL and the Port Instruction
:9210 : Decoder must operate correctly.
:9211 : The CPU first issues a MISC [SET.PORT.I.0] which will assert
:9212 : SEL IDC L.
:9213 : The inputs to the PORT INTERFACE REGISTER PAL on the
:9214 : PORT.READ PORT.ADRS[DAR] instruction will be as follows:
:9215 : CSR 17 H = H
:9216 : CSR 14 H = L
:9217 : CSR 12 H = L
:9218 : CSR 11 H = L
:9219 : CSR 10 H = H
:9220 : PORT INSTR H = H
:9221 : CPU P2 H = H
:9222 : The resultant outputs will be:
:9223 : R2 L = H
:9224 : R1 L = H
:9225 : R0 L = L
:9226 : The CPU then issues a MOV PORT TO LSC[] which asserts READ PORT L which
:9227 : is an input to the PORT INTERFACE REG PAL. This input will cause
:9228 : READ PORT H to assert. READ PORT H is ANDED with SEL IDC H = H and
:9229 : SEL ACC L = H and will assert READ IDC L.
:9230 : READ IDC L and READ PORT H are the enables for the decoder of R<2:0>.
:9231 : R<2:0> will be decoded and will assert SEL DAR L which enables the
:9232 : DAR onto the I/O Bus. READ IDC L also changes the direction of the
:9233 : BUS TRANCEIVERS thus enabling the data from the I/O BUS onto the
:9234 : Y BUS.
:9235 :
:9236 : ERROR 2 -
:9237 :
:9238 : The IDC microcode will branch to instruction 065 which will assert
:9239 : UINCR DAR L. UINCR DAR L is an input to the DAR and will increment the
:9240 : DAR on the next P2 CLOCK L.
:9241 : The CPU will then read the DAR and check the contents.
:9242 :
:9243 : --
:9244 : *****
:9245 : T.9:
:9246 : MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT 15 IN WR[0] FOR
:9247 : ; CONTROL AND STATUS WORD
:9248 : MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND

```

```

U 096C, B65E,15
U 096D, 3E80,15

```

```

:9249 ; STATUS WORD - BIT 15
:9250 ; INDICATES START OF TEST TO
:9251 ; CONSOLE
U 096E, 10E0,15 :9252 MISC [SET.CP.ATTN] ; ISSUE CPU ATTN TO CONSOLE
:9253 WAIT.T9.0: ;
U 096F, 8896,F4 :9254 JMP [WAIT.T9.0] ; LOOP HERE WAITING FOR
:9255 ; CONSOLE TO RESPOND
:9256
U 0970, 65A0,15 :9257 CLR LS[PREVIOUS.ERROR] ; CLEAR PREVIOUS ERROR NUMBER
:9258 ; IN LS
:9259
U 0971, B64A,15 :9260 MOV LS[IDC] TO WR[0] ; STORE MODULE CODE IN LS TO
U 0972, 3E8C,15 :9261 MOV WR[0] TO LS[MODULE.NUM] ; INDICATE IDC MODULE
:9262
:9263
:9264
U 0973, 3776,15 :9265 MOV LS[DAR.MASK] TO WR[0] ; GET THE ERROR MASK TO CHECK
U 0974, 3E8A,15 :9266 MOV WR[0] TO LS[ERROR.MASK] ; BITS <18:0> OF THE DAR
:9267
U 0975, 8872,EC :9268 JSR [DIAG.CODE] ; FORCE THE IDC MICROCODE TO
:9269 ; DIAG.IDLE
U 0976, B795,15 :9270 MOV LS[#7FFFF] TO WR[2] ; FINAL LOOP COUNT
:9271
:9272
U 0977, B640,15 :9273 MOV LS[#1] TO WR[0] ; SET WRO = 1
U 0978, BE82,15 :9274 MOV WR[0] TO LS[ERROR.NUMBER] ; SET UP ERROR NUMBER = 1 IN LS
:9275
U 0979, E510,15 :9276 CLR LS[T8] ; USE LS[T8] FOR EXPECTED DATA
:9277 ; AND LOOP COUNTER
:9278
U 097A, 2F80,15 :9279 LOOP.T9.1: CLR WR[0] ; GET THE DATA TO CLR THE DAR
U 097B, 97A4,15 :9280 PORT.WRITE PORT.ADRS[DAR] WITH WR[0] ; CLEAR THE DAR
:9281
U 097C, 9090,15 :9282 MISC [SET.PORT.I.0] ; SELECT THE IDC
U 097D, 1784,15 :9283 PORT.READ PORT.ADRS[DAR] ; SET UP TO READ THE DAR
U 097E, 3B32,15 :9284 MOV PORT TO LS[PORT0] ; READ THE DATA FROM DAR AND
:9285 ; SAVE IT IN LS
U 097F, 3732,15 :9286 MOV LS[PORT0] TO WR[0] ; SET UP RECEIVED DATA
U 0980, 3610,95 :9287 MOV LS[T8] TO WR[1] ; SET UP EXPECTED DATA
U 0981, 0869,3C :9288 JSR [CHECK.RESULT] ; CHECK THE DATA
U 0982, 0897,A4 :9289 JMP [LOOP.T9.1] ; LOOP HERE IF ERR & LOE IS SET
:9290
U 0983, FF82,15 :9291 INC LS[ERROR.NUMBER] ; SET UP FOR ERROR 2
:9292
U 0984, 8898,84 :9293 JMP [INCR.DAR.T9.1]
:9294
U 0985, 7C10,15 :9295 LOOP.T9.2: DEC LS[T8] ; IF ERROR GO BACK ONE COUNT AND
U 0986, B610,15 :9296 MOV LS[T8] TO WR[0] ; LOAD IT INTO THE DAR
U 0987, 97A4,15 :9297 PORT.WRITE PORT.ADRS[DAR] WITH WR[0]
:9298
:9299
U 0988, 7F10,15 :9300 INCR.DAR.T9.1: INC LS[T8] ; INCREMENT THE CPU COUNTER
:9301
U 0989, B720,95 :9302 MOV LS[SETCSR.MAINT] TO WR[1] ; SET UP THE CSR TO BRANCH TO
U 098A, 3702,15 :9303 MOV LS[SETCSR.F1] TO WR[0] ; THE CORRECT ROUTINE

```

```

U 098B, AEC2,15 :9304 BIS WR[1] TO WR[0] ;CSR F<2:0>=001 MAINT=1
U 098C, 17A0,15 :9305 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:9306 ;MAINT=1,CRDY=0,F<2:0>=001
:9307 -----
:9308 -----
:9309 *****
:9310 * IDC *
:9311 *****
:9312 -----
:9313 018:
:9314 =000
:9315 DIAG.IDLE:
:9316
:9317 BEN0/F0,
:9318 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:9319 BEN2/F2,
:9320 NAD/DIAG.IDLE
:9321 -----
:9322
:9323 019:
:9324 =001
:9325
:9326 BEN0/CRDY.L,
:9327 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:9328 NAD/TEST.FUNC1 ;DEPENDENT ON CSR <CRDY> AND <MAINT>
:9329 -----
:9330
:9331 065:
:9332 =1*1
:9333
:9334 TEST.F1:
:9335 INCR.DAR/ON, ;INCREMENT THE DAR
:9336 NAD/XFER.REQ ;SEND XFER.REQ
:9337 -----
:9338
:9339
:9340 WAIT.IDC.T9.1:
:9341 SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ FROM IDC
:9342 JMP [IDC.DONE.T9.1] ;XFER REQ IS ASSERTED
:9343 JMP [WAIT.IDC.T9.1] ;WAIT SOME MORE
:9344
:9345 IDC.DONE.T9.1:
:9346 MOV LS[SETCSR.F0] TO WR[0] ;SET UP TO CLR CSR F<2:0>
:9347 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:9348 MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
:9349 ; AND FORCE IDC TO DIAG.IDLE
:9350
:9351 READ.DAR.T9.1:
:9352 MISC [SET.PORT.I.0] ;SELECT THE IDC
:9353 PORT.READ PORT.ADRS[DAR] ;SET UP TO READ THE DAR
:9354 MOV PORT TO LS[PORT0] ;READ THE DATA FROM DAR AND
:9355 ;SAVE IT IN LS
:9356 MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
:9357 MOV LS[T8] TO WR[1] ;SET UP EXPECTED DATA
:9358 JSR [CHECK.RESULT] ;CHECK THE DATA
:9359 JMP [LOOP.T9.2] ;LOOP HERE IF ERR & LOE IS SET

```

U 098D, DB00,1A  
 U 098E, 8899,04  
 U 098F, 8898,D4

U 0990, B700,15  
 U 0991, 17A0,15  
 U 0992, 90FA,15

U 0993, 9090,15  
 U 0994, 1784,15  
 U 0995, 3B32,15

U 0996, 3732,15  
 U 0997, 3610,95  
 U 0998, 0869,3C  
 U 0999, 0898,54

ZZ-1  
 :  
 :

U 0

U 0

U 0

U 0

U 0

[illegible]

```

9373 .PAGE "Test A BEN0/CRDY AND BEN2/MAINT BRANCH TEST (M8388 IDC MODULE)"
9374 .**
9375
9376 .Test Description:
9377
9378 .This test is designed to verify that the BEN0/CRDY and BEN2/MAINT
9379 .branches function correctly. If the branch executes correctly, the
9380 .DAR will be incremented. The CPU will then read the DAR to verify that
9381 .it was incremented correctly. To verify that the branch incremented
9382 .correctly the DAR will be incremented a different number of times for
9383 .each data pattern:
9384 .Data pattern 1 - increment the DAR 2 times
9385 .Data pattern 2 - increment the DAR 3 times
9386 .Data pattern 3 - increment the DAR 4 times
9387
9388 .The data patterns are as follows:
9389
9390 .1. CSR <CRDY> = 0, CSR <MAINT> = 0
9391 .2. CSR <CRDY> = 1, CSR <MAINT> = 1
9392 .3. CSR <CRDY> = 1, CSR <MAINT> = 0
9393
9394 .Assumptions:
9395
9396 .It is assumed that the UINCR.DAR function works correctly.
9397
9398 .Test Steps:
9399
9400 .1. Set up error mask, error number and module number in LS
9401 .(for error printouts) and clear the previous error number in LS.
9402 .2. Force the IDC microcode to DIAG.IDLE.
9403 .3. Get the data pattern for CSR <CRDY> <MAINT>.
9404 .4. Clear the DAR.
9405 .5. Set up the CSR with the data pattern.
9406
9407 .*****
9408 .* IDC *
9409 .*****
9410
9411 .IDC1. Increment the DAR (X) number of times depending on the
9412 .data pattern.
9413 .IDC2. Send XFER REQ to flag the CPU.
9414
9415 .6. Wait for XFER.REQ from IDC.
9416 .7. Clear CSR F<2:0> and send XFER GRAN to clear XFER.REQ.
9417 .8. Check the DAR.
9418 .9. Increment the error number.
9419 .10. Get the next data pattern and go to Step 3 until all data patterns
9420 .have been tested.
9421 .11. Clear IDC.
9422 .12. End test.
9423
9424
9425
9426
9427

```

```

:9428 : Errors:
:9429 :
:9430 : Printout:
:9431 : EXPECTED RECEIVED
:9432 : DAR DAR
:9433 :
:9434 : ERROR 1 - The microcode did not branch correctly using data pattern 1.
:9435 :
:9436 : ERROR 2 - The microcode did not branch correctly using data pattern 2.
:9437 :
:9438 : ERROR 3 - The microcode did not branch correctly using data pattern 3.
:9439 :
:9440 : Debug:
:9441 :
:9442 : ERROR 1:
:9443 :
:9444 : The CSR F<2:0> bits will cause the IDC microcode to branch
:9445 : to a word with BEN0/CRDY L and BEN2/MAINT H selected. During
:9446 : this microword the inputs to the branch logic will be as follows:
:9447 :
:9448 : The inputs to the BEN0 branch MUX will be:
:9449 : BEN0 S0 H = H
:9450 : BEN1 S1 H = H
:9451 : BEN2 S2 H = L
:9452 : CRDY L = L
:9453 : The inputs to the BEN2 branch MUX will be:
:9454 : BEN2 S0 H = H
:9455 : BEN2 S1 H = L
:9456 : BEN2 S2 H = H
:9457 : MAINT H = L
:9458 : These conditions will cause the microcode to branch to a
:9459 : routine that will increment the DAR 2 times.
:9460 :
:9461 : ERROR 2:
:9462 :
:9463 : The CSR F<2:0> bits will cause the IDC microcode to branch
:9464 : to a word with BEN0/CRDY L and BEN2/MAINT H selected. During
:9465 : this microword the inputs to the branch logic will be as follows:
:9466 :
:9467 : The inputs to the BEN0 branch MUX will be:
:9468 : BEN0 S0 H = H
:9469 : BEN1 S1 H = H
:9470 : BEN2 S2 H = L
:9471 : CRDY L = H
:9472 : The inputs to the BEN2 branch MUX will be:
:9473 : BEN2 S0 H = H
:9474 : BEN2 S1 H = L
:9475 : BEN2 S2 H = H
:9476 : MAINT H = L
:9477 : These conditions will cause the microcode to branch to a
:9478 : routine that will increment the DAR 3 times.
:9479 :
:9480 :
:9481 : ERROR 3:
:9482 :

```

U OA

U OA

U OA

U OA

U OA

U OA

U OA

U OA

U OA

U OA

U OA

U OA

U OA

U OA

U OA

U OA  
U OA

U 04

1104

U O4

U 04

1

U 04

U 04

U 04

:9483 : The CSR F<2:0> bits will cause the IDC microcode to branch  
 :9484 : to a word with BEN0/CRDY L and BEN2/MAINT H selected. During  
 :9485 : this microword the inputs to the branch logic will be as follows:

The inputs to the BEN0 branch MUX will be:

BEN0 S0 H = H  
 BEN1 S1 H = H  
 BEN2 S2 H = L  
 CRDY L = L

The inputs to the BEN2 branch MUX will be:

BEN2 S0 H = H  
 BEN2 S1 H = L  
 BEN2 S2 H = H  
 MAINT H = H

These conditions will cause the microcode to branch to a routine that will increment the DAR 4 times.

--

\*\*\*\*\*

T.A:

|                 |       |                                      |                                 |
|-----------------|-------|--------------------------------------|---------------------------------|
| U 099D, B65E,15 | :9502 | MOV LS[BEGIN.TEST] TO WR[0]          | ;SET BIT 15 IN WR[0] FOR        |
|                 | :9503 |                                      | ; CONTROL AND STATUS WORD       |
| U 099E, 3E80,15 | :9504 | MOV WR[0] TO LS[CONTROL.STATUS]      | ;SET BIT 15 IN CONTROL AND      |
|                 | :9505 |                                      | ; STATUS WORD - BIT 15          |
|                 | :9506 |                                      | ; INDICATES START OF TEST TO    |
|                 | :9507 |                                      | ; CONSOLE                       |
| U 099F, 10F0,15 | :9508 | MISC [SET.CP.ATTN]                   | ;ISSUE CPU ATTN TO CONSOLE      |
|                 | :9509 | WAIT.TA.0:                           |                                 |
| U 09A0, 889A,04 | :9510 | JMP [WAIT.TA.0]                      | ;LOOP HERE WAITING FOR          |
|                 | :9511 |                                      | ; CONSOLE TO RESPOND            |
|                 | :9512 |                                      |                                 |
| U 09A1, 65A0,15 | :9513 | CLR LS[PREVIOUS.ERROR]               | ;CLEAR PREVIOUS ERROR NUMBER    |
|                 | :9514 |                                      | ; IN LS                         |
|                 | :9515 |                                      |                                 |
| U 09A2, B64A,15 | :9516 | MOV LS[IDC] TO WR[0]                 | ;STORE MODULE CODE IN LS TO     |
| U 09A3, 3E8C,15 | :9517 | MOV WR[0] TO LS[MODULE.NUM]          | ; INDICATE IDC MODULE           |
|                 | :9518 |                                      |                                 |
|                 | :9519 |                                      |                                 |
| U 09A4, B640,15 | :9520 | MOV LS[#1] TO WR[0]                  | ;SET WRO = 1                    |
| U 09A5, BE82,15 | :9521 | MOV WR[0] TO LS[ERROR.NUMBER]        | ;SET UP ERROR NUMBER = 1 IN LS  |
|                 | :9522 |                                      |                                 |
| U 09A6, 3776,15 | :9523 | MOV LS[DAR.MASK] TO WR[0]            | ;ERROR MASK TO CHECK BITS       |
| U 09A7, 3E8A,15 | :9524 | MOV WR[0] TO LS[ERROR.MASK]          | ; <18:0> OF THE DAR             |
|                 | :9525 |                                      |                                 |
| U 09A8, 8872,EC | :9526 | JSR [DIAG.CODE]                      | ;FORCE THE IDC MICROCODE        |
|                 | :9527 |                                      | ; TO DIAG.IDLE                  |
|                 | :9528 | LOOP.TA.1:                           |                                 |
| U 09A9, 2F80,15 | :9529 | CLR WR[0]                            | ;DATA TO CLR THE DAR            |
| U 09AA, 97A4,15 | :9530 | PORT.WRITE PORT.ADRS[DAR] WITH WR[0] | ;WRITE IT                       |
|                 | :9531 |                                      |                                 |
| U 09AB, 3702,15 | :9532 | MOV LS[SETCSR.F1] TO WR[0]           | ;CLR CSR <CRDY> & <MAINT>       |
|                 | :9533 |                                      | ; AND SET THE FNC BITS TO       |
|                 | :9534 |                                      | ; BRANCH TO THE CORRECT ROUTINE |
| U 09AC, 17A0,15 | :9535 | PORT.WRITE PORT.ADRS[CSR] WITH WR[0] | ;WRITE THE CSR                  |
|                 | :9536 |                                      | ;MAINT=0,CRDY=0,F<2:0>=001      |
|                 | :9537 |                                      |                                 |



U O  
U O  
U O  
U O

U OA  
U OA  
U OA  
U OA  
U OA  
U OA  
U OA  
U OA

U OAG  
U OAG

U 0A  
U 0A  
U 0A  
U 0A  
U 0A  
U 0A

U OA  
U OA  
U OA  
U OA  
U OA



U U U U U U U U U

It is assumed that all previous tests have been run successfully.

1. Set up error mask, error number and module number in LS (for error printouts) and clear the previous error number in LS.
2. Force the IDC to DIAG.IDLE.
3. Set CSR <28> to inhibit timeouts, set CSR <MAINT> so the IDC microcode will branch to DIAG.IDLE and clear CSR <CRDY> to start the timer.
4. Set up a loop to wait 400ms.
5. Check CSR <CRDY> = 0. (If the IDC timed out, the IDC will JAM and the microcode will set CSR <CRDY>.)
6. Clear IDC.
7. End test.

Printout:

| EXPECTED<br>CSR | RECEIVED<br>CSR |
|-----------------|-----------------|
|-----------------|-----------------|

ERROR 1 - The IDC timed out with CSR <28> WRITE and TIMEOUT INHIBIT set.

CSR <CRDY> is cleared with CSR <MAINT> set which will cause the IDC microcode to branch to DIAG.IDLE. Clearing CSR <CRDY> also will start the TIMEOUT timer. CSR <28> is set. CSR <28> is gated with the output of the TIMER one-shot and will prohibit the IDC from JAMMING when the one-shot is fired. If the IDC does JAM, the IDC microcode will set CSR <CRDY>.

i.B:

```
;SET BIT 15 IN WR[0] FOR
; CONTROL AND STATUS WORD
;SET BIT 15 IN CONTROL AND
; STATUS WORD - BIT 15
; INDICATES START OF TEST TO
; CONSOLE
```

```

:9804 MOV WR[0] TO LS[CONTROL.STATUS]

```

|              |   |           |                                                           |                                       |              |          |
|--------------|---|-----------|-----------------------------------------------------------|---------------------------------------|--------------|----------|
| ZZ-ENKCF-1.4 | : | ENKCF.MIC | Test B CSR WRITE 115-MAR-83                               | Fiche 2 Frame C2                      | Sequence 221 | Page 215 |
| :            | : | ENKCF.MCR | MICRO2 1M(01) 15-MAR-83 11:46:22                          | ENKCF Micro-diagnostic for IDC module | Rev 01.40    |          |
| :            | : | ENKCF.MIC | Test B CSR WRITE INHIBIT BIT <28> TEST (M8388 IDC MODULE) |                                       |              |          |

  

|                 |       |                                                                         |                                |
|-----------------|-------|-------------------------------------------------------------------------|--------------------------------|
| U 09E7, 10E0,15 | :9808 | MISC [SET.CP.ATTN]                                                      | :ISSUE CPU ATTN TO CONSOLE     |
|                 | :9809 | WAIT.TB.0:                                                              |                                |
| U 09E8, 889E,84 | :9810 | JMP [WAIT.TB.0]                                                         | :LOOP HERE WAITING FOR         |
|                 | :9811 |                                                                         | : CONSOLE TO RESPOND           |
|                 | :9812 |                                                                         |                                |
| U 09E9, 65A0,15 | :9813 | CLR LS[PREVIOUS.ERROR]                                                  | :CLEAR PREVIOUS ERROR NUMBER   |
|                 | :9814 |                                                                         | : IN LS                        |
|                 | :9815 |                                                                         |                                |
| U 09EA, B64A,15 | :9816 | MOV LS[IDC] TO WR[0]                                                    | :STORE MODULE CODE IN LS TO    |
| U 09EB, 3E8C,15 | :9817 | MOV WR[0] TO LS[MODULE.NUM]                                             | : INDICATE IDC MODULE          |
|                 | :9818 |                                                                         |                                |
|                 | :9819 |                                                                         |                                |
| U 09EC, B640,15 | :9820 | MOV LS[#1] TO WR[0]                                                     | :SET WRO = 1                   |
| U 09ED, BE82,15 | :9821 | MOV WR[0] TO LS[ERROR.NUMBER]                                           | :SET UP ERROR NUMBER = 1 IN LS |
|                 | :9822 |                                                                         |                                |
| U 09EE, B782,15 | :9823 | MOV LS[BIT7.MASK] TO WR[0]                                              |                                |
| U 09EF, 3E8A,15 | :9824 | MOV WR[0] TO LS[ERROR.MASK]                                             | :SET UP MASK TO CHECK BIT <7>  |
|                 | :9825 |                                                                         |                                |
|                 | :9826 |                                                                         |                                |
|                 | :9827 | LOOP.TB.1:                                                              |                                |
| U 09F0, 17CC,15 | :9828 | PORT.CONTROL [CLEAR.IDC]                                                | :JAM THE IDC TO IDLE           |
| U 09F1, 17CC,15 | :9829 | PORT.CONTROL [CLEAR.IDC]                                                |                                |
| U 09F2, 17CC,15 | :9830 | PORT.CONTROL [CLEAR.IDC]                                                |                                |
|                 | :9831 |                                                                         |                                |
| U 09F3, 3720,15 | :9832 | MOV LS[SETCSR.MAINT] TO WR[0]                                           | :SET UP THE CSR..SET MAINT,    |
|                 | :9833 |                                                                         | : CLEAR CRDY                   |
| U 09F4, B678,95 | :9834 | MOV LS[BIT28] TO WR[1]                                                  | :SET WRITE INHIBIT             |
|                 | :9835 |                                                                         |                                |
| U 09F5, AEC2,15 | :9836 | BIS WR[1] TO WR[0]                                                      |                                |
| U 09F6, 17A0,15 | :9837 | PORT.WRITE PORT.ADRS[CSR] WITH WR[0]                                    | :MAINT=1,CRDY=0,INHIBIT=1      |
|                 | :9838 |                                                                         |                                |
|                 | :9839 | -----                                                                   |                                |
|                 | :9840 | :THE IDC MICROCODE WILL BRANCH FROM THE IDLE LOOP TO DIAG.IDLE AND WILL |                                |
|                 | :9841 | :REMAIN AT DIAG.IDLE UNTIL THE CLEANUP ROUTINE IS EXECUTED              |                                |
|                 | :9842 | -----                                                                   |                                |
| U 09F7, 37E6,15 | :9843 | MOV LS[#B4D84] TO WR[0]                                                 | :SET UP A LOOP TO WAIT 400ms   |
|                 | :9844 | WAIT.LOOP.TB:                                                           |                                |
|                 | :9845 | DEC WR[0]                                                               |                                |
| U 09F8, A100,35 | :9846 | DT(LONG)&SET.ALU.CC                                                     |                                |
| U 09F9, 089F,81 | :9847 | JMP.IF[NEQ] TO [WAIT.LOOP.TB]                                           | :WAIT 400ms                    |
|                 | :9848 |                                                                         |                                |
|                 | :9849 | READ.CSR.TB.1:                                                          |                                |
| U 09FA, 9090,15 | :9850 | MISC [SET.PORT.I.0]                                                     | :SELECT THE IDC                |
| U 09FB, 9780,15 | :9851 | PORT.READ PORT.ADRS[CSR]                                                | :SET UP TO READ THE CSR        |
| U 09FC, 3B32,15 | :9852 | MOV PORT TO LS[PORT0]                                                   | :READ THE DATA FROM CSR AND    |
|                 | :9853 |                                                                         | : SAVE IT IN LS                |
| U 09FD, 3732,15 | :9854 | MOV LS[PORT0] TO WR[0]                                                  | :SET UP RECEIVED DATA          |
| U 09FE, 2F82,95 | :9855 | CLR WR[1]                                                               | :SET UP EXPECTED DATA          |
| U 09FF, 0869,3C | :9856 | JSR [CHECK.RESULT]                                                      | :CHECK THE DATA                |
| U 0A00, 889F,04 | :9857 | JMP [LOOP.TB.1]                                                         | :LOOP HERE IF ERR & LOE IS SET |
|                 | :9858 |                                                                         |                                |
| U 0A01, 8875,3C | :9859 | JSR [CLEANUP]                                                           | :CLEANUP THE CONDITIONS SET BY |
|                 | :9860 |                                                                         | : THIS TEST                    |
|                 | :9861 | END.TB:                                                                 | :END TEST                      |
|                 | :9862 |                                                                         |                                |

22-ENKCF-1.4 ; ENKCF.MIC Test B CSR WRITE 115-MAR-83 D 2 Fiche 2 Frame D2 Sequence 222  
: ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40 Page 216  
: ENKCF.MIC Test B CSR WRITE INHIBIT BIT <28> TEST (M8388 IDC MODULE)  
;9863

22-  
:  
:

U  
U  
U  
U

```

:9864 .PAGE 'Test C DRIVE READY BRANCH TEST (M8388 IDC MODULE)'
:9865 :++
:9866
:9867 Test Description:
:9868
:9869 This test is designed to verify the BENO/DRIVE RDY H branch condition.
:9870 Drive 0 will be selected from the CSR bits <9:8>. BENO/DRIVE RDY will
:9871 be selected from the IDC microcode and if the branch condition is
:9872 asserted, then the DAR will be incremented. The CSR and the DAR will
:9873 then be read. If the drive is ready, CSR <0> (DRV RDY) will = 1 and
:9874 the DAR should = 1. If the drive is not ready then CSR <0> (DRV RDY)
:9875 will = 0 and DAR should = 0. The next drive will then be selected and
:9876 the above sequence repeated for drives <3:0>.
:9877
:9878 Assumptions:
:9879
:9880 It is assumed that the UINCR.DAR function works properly and the CSR
:9881 branch conditions, F<2:0>, <CRDY> AND <MAINT> that will allow the
:9882 microcode to branch to the correct routine, work properly.
:9883
:9884 Test Steps:
:9885
:9886 1. Set up error mask, error number and module number in LS
:9887 (for error printouts) and clear the previous error number in LS.
:9888 2. Force the IDC microcode to DIAG.IDLE.
:9889 3. Select Drive 0 from the CSR <9:8>.
:9890 4. Clear the DAR.
:9891 5. Set up the CSR to branch to the correct IDC microcode routine.
:9892
:9893 *****
:9894 * IDC *
:9895 *****
:9896
:9897 IDC1. Select BENO/DRIVE RDY H.
:9898 IDC2. If DRIVE RDY H is asserted, then increment the DAR.
:9899 IDC3. Send XFER REQ to flag the CPU.
:9900
:9901 6. Wait for a XFER.REQ from the IDC.
:9902 7. Clear CSR FNC <2:0> and issue XFER GRANT to clear the XFER.REQ.
:9903 8. If CSR bit <0> = 1 then check that the DAR = 1.
:9904 9. If CSR bit <0> = 0 then check that the DAR = 0.
:9905 10. Select the next drive from the CSR <9:8> and Repeat Steps 4 - 10
:9906 until drives <3:0> have been tested.
:9907 11. Clear IDC.
:9908 12. End test.
:9909
:9910 Errors:
:9911
:9912 ERROR 1 - CSR bit 0 failed or BENO/DRIVE RDY H failed.
:9913
:9914 Printout:
:9915 EXPECTED RECEIVED OTHER DATA
:9916 DAR DAR DRIVE NUMBER
:9917 LS[T8]
:9918

```



```

9919
9920
9921
9922 Debug:
9923
9924 ERROR 1:
9925
9926 The Drive number under test will be selected by CSR <9:8>. These bits
9927 are inputs to a FF and are latched with P2 CLOCK L when enabled by
9928 WRITE CSR L = L. These bits are output from the FF as DS1 H and DS0 H.
9929 DS1 H and DS0 H are inputs to the DR SEL LOG PAL. USEL DRV H will be
9930 L so DS1 H is output as DRIVE SEL 1 H and DS0 H is output as DRIVE
9931 SEL 0 H.
9932 If the drive selected is an RLO2, then DRIVE SEL <1:0> are
9933 driven off the module, to the disk drive. The selected drive will
9934 return its ready state as RL DRIVE READY H. This signal is input to the
9935 DR SEL LOG PAL and will be output as DRV RDY when RLO2 H = H.
9936 If the drive selected is an R80, then RLO2 H = L. The output DRV RDY H
9937 in this case is determined by two input signals. If either R80 ON
9938 CYLINDER H = L or R80 DRIVE READY H = L, then DRV RDY H = L.
9939 DRV RDY H is latched by P2 CLOCK L in the CSR SELECTED STATUS LATCH
9940 PAL. When the CSR is read L DRV RDY H is output as BUS IO 00 H.
9941 DRV RDY H is also an input to the INIT FF and is latched by P2 CLOCK
9942 L. It is output as DRIVE RDY H.
9943 In the microstate which selects BENO/DRIVE.RDY, the outputs of the
9944 microcode PROMs should be:
9945 BENO S0 H = 1
9946 BENO S1 H = 0
9947 BENO S2 H = 0
9948
9949 These signals are the selects for the branch logic and will select
9950 DRIVE RDY H which becomes NUA 0 H. If DRIVE RDY H was asserted, then
9951 the microcode will branch to an instruction that will increment the
9952 DAR.
9953 The CPU will read the CSR and if bit 0 is set, then the expected data
9954 for the DAR is 1 since this indicates that the branch condition should
9955 have been asserted.
9956 If CSR bit 0 is clear, then the expected data for the DAR is 0 since
9957 this indicates that the branch condition should not have been asserted.
9958
9959
9960
9961
9962
9963
9964
9965
9966
9967
9968
9969
9970
9971
9972
9973

```

[illegible]

|   |   |
|---|---|
| U | O |
| U | O |
|   |   |
| U | O |
| U | O |
|   |   |
| U | O |
| U | O |
|   |   |
| U | O |
| U | C |
| U | C |
|   |   |
| U | C |
| U | C |
|   |   |
| U | C |
| U | C |
|   |   |
| U | C |
|   |   |
| U | ( |
| U | ( |
|   |   |
| U | ( |
|   |   |
| U | ( |

U 0A1D. 8700.15  
U 0A1E. 17A0.15

```

U 0A1F, 90FA,15 :10084 MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR XFER REQ
 :10085
 :10086
U 0A20, 9090,15 :10087 READ.CSR.TC.1:
U 0A21, 9780,15 :10088 MISC [SET.PORT.I.0] ;SELECT THE IDC
U 0A22, 3B32,15 :10089 PORT.READ PORT.ADRS[CSR] ;SET UP TO READ THE CSR
 :10090 MOV PORT TO LS[PORT0] ;READ THE DATA FROM CSR AND
 :10091 MOV LS[PORT0] TO WR[0] ;SAVE IT IN LS
U 0A23, 3732,15 :10092
 :10093 CLR WR[1]
U 0A24, 2F82,95 :10094
 :10095 BIT LS[BIT0] WITH WR[0],
 :10096 DT(LONG)&SET.ALU.CC ;CLR THE EXPECTED DATA
 :10097 JMP.IF[BITS.CLR] TO [READ.DAR.TC.1] ;IS DRV RDY SET?
U 0A25, 5940,35 :10098 MOV LS[#1] TO WR[1] ;JMP IF DRV NOT READY
U 0A26, 08A2,89 :10099
 :10100 READ.DAR.TC.1:
 :10101 MISC [SET.PORT.I.0] ;SELECT THE IDC
U 0A27, 3640,95 :10102 PORT.READ PORT.ADRS[DAR] ;SET UP TO READ THE DAR
 :10103 MOV PORT TO LS[PORT0] ;READ THE DATA FROM DAR AND
 :10104 MOV LS[T8] TO WR[0] ;SAVE IT IN LS
U 0A28, 9090,15 :10105 MOV WR[0] TO LS[ADDRESS.DATA] ;GET THE DRIVE NUMBER FOR
U 0A29, 1784,15 :10106 MOV LS[PORT0] TO WR[0] ;PRINTOUT
U 0A2A, 3B32,15 :10107 JSR [CHECK.RESULT] ;SET UP RECEIVED DATA
 :10108 JMP [LOOP.TC.1] ;CHECK THE DATA
U 0A2B, B610,15 :10109
 :10110 MOV LS[#100000] TO WR[0] ;LOOP HERE IF ERR & LOE IS SET
U 0A2C, BE88,15 :10111 ADD WR[0] TO WR[2]
U 0A2D, 3732,15 :10112 INC LS[T8]
U 0A2E, 0869,3C :10113
 :10114 DEC WR[3],
 :10115 DT(LONG)&SET.ALU.CC ;INCREMENT THE DRIVE NUMBER
U 0A2F, 08A1,34 :10116 JMP.IF[NEQ] TO [TEST.DRV.RDY.TC] ;INC THE DRIVE NUMBER FOR
 :10117
 :10118 JSR [CLEANUP]
U 0A30, B668,15 :10119
 :10120 END.TC:
U 0A31, AE01,15 :10121
 :10122
U 0A32, 7F10,15 :10123
 :10124
U 0A33, A107,B5 :10124
U 0A34, 08A1,31 :
U 0A35, 8875,3C :

```

10125 : PAGE "Test D XFER.REQ BRANCH TEST (M8388 IDC MODULE)"

10126 : \*\*

10127 :

10128 :

10129 :

10130 :

10131 :

10132 :

10133 :

10134 :

10135 :

10136 :

10137 :

10138 :

10139 :

10140 :

10141 :

10142 :

10143 :

10144 :

10145 :

10146 :

10147 :

10148 :

10149 :

10150 :

10151 :

10152 :

10153 :

10154 :

10155 :

10156 :

10157 :

10158 :

10159 :

10160 :

10161 :

10162 :

10163 :

10164 :

10165 :

10166 :

10167 :

10168 :

10169 :

10170 :

10171 :

10172 :

10173 :

10174 :

10175 :

10176 :

10177 :

10178 :

10179 :

# Test Description:

This test is designed to insure that BEN1/XFER REQ H branch executes correctly. The IDC will first assert XFER REQ. The IDC microcode will then execute a BEN1/XFER REQ H and if XFER REQ is asserted, an instruction to increment the DAR will be executed. The CPU will then check the contents of the DAR. The CPU will then issue XFER GRANT to clear the XFER.REQ. A BEN1/XFER REQ H branch will then be executed and if XFER REQ has not deasserted, the DAR will be incremented. The CPU will then check the contents of the DAR.

## Assumptions:

It is assumed that the UINCR.DAR function works correctly.

## Test Steps:

1. Set up error mask, error number, and module number in LS (for error printouts) and clear previous error number in LS.
2. Force the IDC microcode to DIAG.IDLE.
3. Clear the DAR.
4. Set up the CSR to branch to the correct microcode routine.
5. NOP to allow time for the IDC to complete.
6. Clear CSR F<2:0>.

\*\*\*\*\*

\* IDC \*

\*\*\*\*\*

- IDC1. Execute an instruction with UCM = 4, SET XFER.REQ.
- IDC2. Select BEN1/XFER REQ H.
- IDC3. If XFER REQ H is asserted, then increment the DAR.
- IDC4. Go to DIAG.WAIT.

7. Check the contents of the DAR = 1.
8. Increment the error number.
9. Clear the DAR.
10. Send XFER GRANT.
11. Set up the CSR to branch to the correct microcode routine.
12. NOP to allow the IDC time to complete.
13. Clear CSR F<2:0>.

\*\*\*\*\*

\* IDC \*

\*\*\*\*\*

- IDC5. Select BEN1/XFER REQ H.
- IDC6. If XFER REQ H is asserted, then increment the DAR.
- IDC7. Go to DIAG.WAIT.

14. Check the contents of the DAR = 0.

```

10180 : 15. Clear IDC.
10181 : 16. End Test.
10182 :
10183 : Error:
10184 :
10185 : Printout:
10186 : EXPECTED RECEIVED
10187 : DAR DAR
10188 :
10189 :
10190 : ERROR 1 - BEN1/XFER REQ H failed asserted.
10191 :
10192 : ERROR 2 - BEN1/XFER REQ H failed unasserted.
10193 :
10194 :
10195 : Debug:
10196 :
10197 : ERROR 1 -
10198 : A microinstruction with UCMD/SET.XFER.RQST is executed. This will
10199 : assert XFER REQ H. The IDC microcode will select BEN1/XFER.REQ.
10200 : The branch select outputs will be:
10201 : BEN1 S0 H = L
10202 : BEN1 S1 H = H
10203 : BEN1 S2 H = L
10204 : This will select XFER REQ H = H as NUA 1 H = H. The microcode will
10205 : branch to an instruction that will increment the DAR. The CPU will
10206 : then read and check the DAR = 1.
10207 :
10208 : ERROR 2 -
10209 : A MISC2 [TRANSFER.GRANT] instruction is executed. This instruction
10210 : will be decoded on the CPU module by the MISC INST DECODER and will
10211 : assert XFER GRANT L.
10212 :
10213 : XFER GRANT L is an input to the CONTROL STATUS REG PAL.
10214 : At the next P2 CLOCK, the output XFER REQ H will deassert.
10215 : The IDC microcode will select BEN1/XFER.REQ. The branch select
10216 : outputs will be:
10217 : BEN1 S0 H = L
10218 : BEN1 S1 H = H
10219 : BEN1 S2 H = L
10220 : This will select XFER REQ H = L as NUA 1 H = L. The microcode will
10221 : will NOT increment the DAR. The CPU will then read and check the
10222 : DAR = 0.
10223 :
10224 :
10225 : *****
10226 : T.D: MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT 15 IN WR[0] FOR
10227 : MOV WR[0] TO LS[CONTROL.STATUS] ;CONTROL AND STATUS WORD
10228 : ;SET BIT 15 IN CONTROL AND
10229 : ;STATUS WORD - BIT 15
10230 : ;INDICATES START OF TEST TO
10231 : ;CONSOLE
10232 : MISC [SET.CP.ATTN] ;ISSUE CPU ATTN TO CONSOLE
10233 : WAIT.TD.0:
10234 : JMP [WAIT.TD.0] ;LOOP HERE WAITING FOR

```

|   |   |
|---|---|
| U | O |
| U | O |
|   |   |
| U | O |
| U | O |
|   |   |
| U | O |
| U | O |
|   |   |
| U | O |
| U | O |
|   |   |
| U | O |
|   |   |
|   |   |





U OE  
U OE  
U OE

U OE  
U OE  
U OE

U OE  
U OE

U OE

```

:10400 :TEST.XFER.REQ.BR.2:
:10401 : NAD/DIAG.IDLE ;XFER REQ DID NOT ASSERT
:10402 :-----
:10403 :18E:
:10404 :=1*
:10405 :TEST.XFER.REQ.BR.3:
:10406 : INCR.DAR/ON, ;XFER REQ ASSERTED
:10407 : NAD/DIAG.IDLE ;INCR THE DAR
:10408 :-----
U OA65, B700,15 :10409 : MOV LS[SETCSR.F0] TO WR[0] ;CLR THE FUNC BITS BEFORE
U OA66, 17A0,15 :10410 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ; RETURN TO DIAG.IDLE
U OA67, DB00,15 :10411 : NOP
U OA68, DB00,15 :10412 : NOP
U OA69, DB00,15 :10413 : NOP
:10414 :READ.DAR.TD.2:
U OA6A, 9090,15 :10415 : MISC [SET.PORT.1.0] ;SELECT THE IDC
U OA6B, 1784,15 :10416 : PORT.READ PORT.ADRS[DAR] ;SET UP TO READ THE DAR
U OA6C, 3B32,15 :10417 : MOV PORT TO LS[PORT0] ;READ THE DATA FROM DAR AND
:10418 : ; SAVE IT IN LS
U OA6D, 3732,15 :10419 : MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
U OA6E, 2F82,95 :10420 : CLR WR[1] ;SET UP EXPECTED DATA
U OA6F, 0869,3C :10421 : JSR [CHECK.RESULT] ;CHECK THE DATA
U OA70, 88A5,C4 :10422 : JMP [LOOP.TD.2] ;LOOP HERE IF ERR & LOE IS SET
U OA71, 8875,3C :10423 : JSR [CLEANUP] ;CLEAN UP THE CONDITIONS SET
:10424 : ; BY THIS TEST
:10425 :END.TD: ;END TEST
:10426
:10427
:10428
:10429
:10430
:10431

```

10432 .PAGE "TEST E DISK DRIVE SIZING TEST (M8388 IDC MODULE)"

10433 :++

10434 :  
10435 : Test Description:

10436 : This test is designed determine if any R80 drives are attached.  
 10437 : If an R80 is found, a message indicating the Drive number of the  
 10438 : R80 will be printed. If no R80s are found, a message indicating this  
 10439 : will be printed. This information will be stored in LS[DRIVE.STAUS]  
 10440 : for use in future tests. Each drive will be selected, and  
 10441 : CSR bit 26 <RL02> will be checked to see what type of a drive  
 10442 : was found.  
 10443 :

10444 :  
10445 : Assumptions:

10446 : It is assumed that all previous test have run successfully.  
 10447 :  
 10448 :

10449 : Test Steps:

- 10450 : 1. Set up error mask, error number, and module number in LS
- 10451 : (for error printouts) and clear the previous error number in LS.
- 10452 : 2. Clear LS[T6.SUB] and LS[T7.SUB]. LS[T6.SUB] will be used to save the
- 10453 : drive number, and LS[T7.SUB] =1 will indicate an R80 was found for
- 10454 : printouts.
- 10455 : 3. Clear LS[T8] and LS[T9] to save drive information.
- 10456 : 4. Write the CSR with the drive number from LS[T6.SUB].
- 10457 : 5. Read the CSR.
- 10458 : 6. If LS[SPECIAL] is set then force an error for looping purposes.
- 10459 : 7. Increment the error number.
- 10460 : 8. If the drive was not an R80 then go to Step 12.
- 10461 : 9. Print message 'R80 PRESENT DRIVE #'.
- 10462 : 10. If this was not the first R80 found then issue error.
- 10463 : 11. Save the Drive information in LS[T8] and LS[T9].
- 10464 : 12. If the last drive has not been tested, then increment the error
- 10465 : number, increment the drive number and go to Step 4.
- 10466 : 13. If no R80s were found then print message, 'R80 NOT PRESENT'.
- 10467 : 14. If APT configuration is not equal to the configuration found
- 10468 : by this test then issue error. Do not validate LS[DRIVE.STATUS].
- 10469 : Go to end of test.
- 10470 : 15. Validate LS[DRIVE.STATUS] and save the drive information.
- 10471 : 16. End test.
- 10472 :

10473 :  
10474 : Errors:

10475 :  
10476 : ERROR 1 - Forced error for looping purposes if the configuration  
 10477 : found by this test does not match actual configuration.  
 10478 :

10479 : Printout:

| 10480 : | 10481 : EXPECTED | 10482 : RECEIVED | 10483 : OTHER DATA   |
|---------|------------------|------------------|----------------------|
| 10484 : | 10485 : WR[1]    | 10486 : WR[0]    | 10487 : DRIVE NUMBER |

10488 : ERROR 2 - More than one R80 was found.

10489 :  
10490 : Printout:

```

:10487 :
:10488 :
:10489 :
:10490 :
:10491 : ERROR 3 - The configuration found was not the same as the one given
:10492 : by APT.
:10493 :
:10494 : Printout:
:10495 : EXPECTED RECEIVED
:10496 : APT CONFIG CONFIG FOUND
:10497 :
:10498 :
:10499 : Debug:
:10500 :
:10501 : LOOPING -
:10502 : To loop on a forced error, SET SP(ECIAL) and run the test.
:10503 : This will force an error. Continue to the error on which you wish to
:10504 : loop, and then type LOOP.
:10505 :
:10506 : ERROR 1 -
:10507 : The drive number will be selected by CSR <9:8>. These bits are latched
:10508 : by P2 CLOCK L and are output as DS1 H and DS0 H. These signals are
:10509 : inputs to the DR SEL LOG PAL. If the drive selected is an R80, then
:10510 : its address will be on R80 SELECT ADRS 1 H and R80 SELECT ADRS 0 H,
:10511 : and PLUG VALID H = H. This will force RL02 H = L.
:10512 : If the drive selected is not an R80, RL02 H = H.
:10513 : RL02 H is latch by P2 CLOCK L in the CSR SELECTED STATUS LATCH PAL and
:10514 : output as L RL02 H. This signal is output as BUS IO 26 H when the
:10515 : CSR is read.
:10516 :
:10517 : ERROR 2 -
:10518 : Since only one R80 is allowed on an IDC module, if this test finds
:10519 : more than one, an error is reported. The Drive number reported as
:10520 : OTHER DATA is the number of the second R80 found. First determine
:10521 : which drive is being reported incorrectly, and then see LOOPING and
:10522 : ERROR 1.
:10523 :
:10524 : ERROR 3 -
:10525 : APT will report the configuration that it expects to be testing.
:10526 : First determine if the APT configuration is wrong or if this test
:10527 : found the wrong configuration. If this test reported the wrong
:10528 : configuration, then see LOOPING and ERROR 1.
:10529 : --
:10530 : *****
:10531 : T.E:
U 0A72, B65E,15 :10532 : MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT 15 IN WR[0] FOR
:10533 : ; CONTROL AND STATUS WORD
U 0A73, 3E80,15 :10534 : MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND
:10535 : ; STATUS WORD - BIT 15
:10536 : ; INDICATES START OF TEST TO
:10537 : ; CONSOLE
U 0A74, 10E0,15 :10538 : MISC [SET.CP.ATTN] ;ISSUE CPU ATTN TO CONSOLE
:10539 : WAIT.TE.0:
U 0A75, 08A7,54 :10540 : JMP [WAIT.TE.0] ;LOOP HERE WAITING FOR
:10541 : ; CONSOLE TO RESPOND

```

```

U 0A76, 65A0,15 :10542 CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
 :10543 ; IN LS
 :10544
 :10545
U 0A77, B64A,15 :10546 MOV LS[IDC] TO WR[0] ;STORE MODULE CODE IN LS TO
U 0A78, 3E8C,15 :10547 MOV WR[0] TO LS[MODULE.NUM] ; INDICATE IDC MODULE
 :10548
 :10549
 :10550
 :10551
U 0A79, E58A,15 :10552 CLR LS[ERROR.MASK] ;SET UP ERROR MASK TO CHECK
 :10553 ; ALL BITS
U 0A7A, 650C,15 :10554 CLR LS[T6.SUB] ;DRV NUM FOR PRINTOUT
U 0A7B, E50E,15 :10555 CLR LS[T7.SUB] ;USED IF R80 FOUND FOR PRINTOUT
U 0A7C, E510,15 :10556 CLR LS[T8] ;STORES R80 , FO
U 0A7D, 6512,15 :10557 CLR LS[T9] ;STORE DRIVE NUM
U 0A7E, 2F85,15 :10558 CLR WR[2] ;STORES THE DRIVE NUMBER
U 0A7F, 2F80,15 :10559 CLR WR[0] ;CLR THE DRIVE STATUS LOCATION
U 0A80, 3F8E,15 :10560 MOV WR[0] TO LS[DRIVE.STATUS] ; IN LS
 :10561
U 0A81, 8872,EC :10562 JSR [DIAG.CODE] ;FORCE THE IDC TO DIAG.IDLE
 :10563
 :10564 TEST.DRIVE.TE.1:
 :10565 LOOP.TE.1:
U 0A82, B640,15 :10566 MOV LS[#1] TO WR[0] ;SET WRO = 1
U 0A83, BE82,15 :10567 MOV WR[0] TO LS[ERROR.NUMBER] ;SET UP ERROR NUMBER = 1 IN LS
 :10568
U 0A84, 2004,15 :10569 MOV WR[2] TO WR[0] ;GET THE DRIVE NUMBER
U 0A85, 3646,95 :10570 MOV LS[#8] TO WR[1] ;SHIFT THE DRIVE NUMBER INTO
 :10571 SHIFT.TE.1:
U 0A86, 23D0,15 :10572 ASHL WR[0] ; BITS <9:8> FOR THE WRITE TO
 :10573 DEC WR[1], ; THE CSR
U 0A87, A102,B5 :10574 DT(LONG)&SET.ALU.CC
U 0A88, 08A8,61 :10575 JMP.IF[NEQ] TO [SHIFT.TE.1]
 :10576 ;DRIVE NUM IS NOW IN BITS<9:8>
 :10577
U 0A89, 4778,15 :10578 BIS LS[BIT28] TO WR[0] ;SET THE TIMEOUT INHIBIT BIT
U 0A8A, 17A0,15 :10579 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
 :10580
U 0A8B, 9090,15 :10581 MISC [SET.PORT.I.O] ;SELECT THE IDC
U 0A8C, 9780,15 :10582 PORT.READ PORT.ADRS[CSR] ;SET UP TO READ THE CSR
U 0A8D, 3B32,15 :10583 MOV PORT ^O LS[PORT0] ;READ THE DATA FROM CSR AND
 :10584 ; SAVE IT IN LS
 :10585
U 0A8E, 364E,15 :10586 MOV LS[SPECIAL] TO WR[0] ;CHECK IF SPECIAL LOOP IS SET
 :10587 BIT LS[BIT0] WITH WR[0],
U 0A8F, 5940,35 :10588 DT(LONG)&SET.ALU.CC
U 0A90, 88A9,89 :10589 JMP.IF[BIT0.CLR] TO [NO.LOOP.TE.1] ;JUMP IF NOT SET
 :10590
U 0A91, B670,15 :10591 MOV LS[OTHER.DATA] TO WR[0] ;SET UP TO PRINT OTHER DATA
U 0A92, 3E80,15 :10592 MOV WR[0] TO LS[CONTROL.STATUS] ;
 :10593
U 0A93, 3E89,15 :10594 MOV WR[2] TO LS[ADDRESS.DATA] ;DRIVE NUMBER UNDER TEST
 :10595
U 0A94, B640,15 :10596 MOV LS[#1] TO WR[0] ;FORCE AN ERROR TO ALLOW

```

```

U 0A95, 2F82,95 :10597 CLR WR[1] ; LOOPING
U 0A96, 0869,3C :10598 JSR [CHECK.RESULT]
U 0A97, 88A8,24 :10599 JMP [LOOP.TE.1] ; LOOP HERE IF ERR & LOE IS SET
:10600
:10601 NO.LOOP.TE.1:
U 0A98, 3642,15 :10602 MOV LS[#2] TO WR[0]
U 0A99, BE82,15 :10603 MOV WR[0] TO LS[ERROR.NUMBER] ; ERROR 2
U 0A9A, 3732,15 :10604 MOV LS[PORT0] TO WR[0]
:10605 BIT LS[BIT26] WITH WR[0], ; WAS THE DRIVE AN R80?
U 0A9B, D974,35 :10606 DT(LONG)&SET.ALU.CC
U 0A9C, 08AB,89 :10607 JMP.IF[BIT5.CLR] TO [RL02.TE.1] ; JUMP IF DRIVE IS NOT AN R80
:10608
U 0A9D, B640,15 :10609 MOV LS[#1] TO WR[0]
U 0A9E, 3E0E,15 :10610 MOV WR[0] TO LS[T7.SUB] ; INDICATES AN R80 FOUND
U 0A9F, 3E0D,15 :10611 MOV WR[2] TO LS[T6.SUB] ; DRIVE NUMBER FOR PRINTOUT
U 0AA0, 364E,15 :10612 MOV LS[PRINT.R80] TO WR[0] ; SET UP TO PRINT THAT AN
U 0AA1, 3E80,15 :10613 MOV WR[0] TO LS[CONTROL.STATUS] ; R80 WAS FOUND
:10614
U 0AA2, 10E0,15 :10615 MISC [SET.CP.ATTN] ; ISSUE CPU ATTN TO CONSOLE
:10616 WAIT.TE.1:
U 0AA3, 88AA,34 :10617 JMP [WAIT.TE.1] ; TO GO PRINT MESSAGE
:10618 ; LOOP HERE WAITING FOR
:10619 ; CONSOLE TO RESPOND
:10620
U 0AA4, B670,15 :10621 MOV LS[OTHER.DATA] TO WR[0] ; SET UP TO PRINT OTHER DATA
U 0AA5, 3E80,15 :10622 MOV WR[0] TO LS[CONTROL.STATUS] ;
:10623
U 0AA6, 3E89,15 :10624 MOV WR[2] TO LS[ADDRESS.DATA] ; DRIVE NUMBER UNDER TEST
:10625
U 0AA7, B610,15 :10626 MOV LS[T8] TO WR[0] ; CHECK IF THIS IS FIRST R80
:10627 ; FOUND
U 0AA8, 2F82,95 :10628 CLR WR[1] ; EXPECTED DATA
U 0AA9, 0869,3C :10629 JSR [CHECK.RESULT] ; CHECK THE DATA
U 0AAA, 88A8,24 :10630 JMP [LOOP.TE.1] ; LOOP HERE IF ERR & LOE IS SET
:10631
U 0AAB, B640,15 :10632 MOV LS[#1] TO WR[0]
U 0AAC, 3E10,15 :10633 MOV WR[0] TO LS[T8] ; SET FOR FIRST R80 FOUND
:10634
U 0AAD, 3640,95 :10635 MOV LS[BIT0] TO WR[1]
:10636 CMP LS[ZERO] WITH WR[2],
U 0AAE, 4E9D,35 :10637 DT(LONG)&SET.ALU.CC
U 0AAF, 08AB,79 :10638 JMP.IF[EQL] TO [R80.SAV.TE] ; JUMP IF DRIVE 0
:10639
U 0AB0, B642,95 :10640 MOV LS[BIT1] TO WR[1]
:10641 CMP LS[#1] WITH WR[2],
U 0AB1, CE41,35 :10642 DT(LONG)&SET.ALU.CC
U 0AB2, 08AB,79 :10643 JMP.IF[EQL] TO [R80.SAV.TE] ; JUMP IF DRIVE 1
:10644
U 0AB3, B644,95 :10645 MOV LS[BIT2] TO WR[1]
:10646 CMP LS[#2] WITH WR[2],
U 0AB4, 4E43,35 :10647 DT(LONG)&SET.ALU.CC
U 0AB5, 08AB,79 :10648 JMP.IF[EQL] TO [R80.SAV.TE] ; JUMP IF DRIVE 2
:10649
U 0AB6, 3646,95 :10650 MOV LS[BIT3] TO WR[1] ; SAVE DRIVE 3 IN WR[1]
:10651

```

```

:10652 R80.SAV.TE:
U OAB7, 6D12,95 :10653 BIS WR[1] TO LS[T9] ;SAVE THIS INFO IN LS
:10654
:10655 RLO2.TE.1:
U OAB8, B788,15 :10656 MOV LS[#3] TO WR[0] ;LAST DRIVE BEEN TESTED?
:10657 CMP WR[0] WITH WR[2],
U OAB9, 2641,35 :10658 DT(LONG)&SET.ALU.CC
U OABA, 88AB,F9 :10659 JMP.IF[EQL] TO [DONE.TE.1] ;JMP IF ALL DRVS BEEN TESTED
U OABB, B640,15 :10660 MOV LS[#1] TO WR[0]
U OABC, BE82,15 :10661 MOV WR[0] TO LS[ERROR.NUMBER] ;ERROR 1
U OABD, 2045,15 :10662 INC WR[2] ;INCREMENT THE DRIVE NUMBER
U OABE, 88A8,24 :10663 JMP [TEST.DRIVE.TE.1]
:10664
:10665 DONE.TE.1:
:10666
U OABF, B796,15 :10667 MOV LS[#F] TO WR[0] ;ANY R80'S FOUND?
:10668 BIT LS[T9] WITH WR[0],
U OAC0, D912,35 :10669 DT(LONG)&SET.ALU.CC
U OAC1, 08AC,81 :10670 JMP.IF[BIT.SET] TO [SAVE.INFO.TE.1] ;JMP IF R80
:10671 CLR WR[0]
U OAC2, 2F80,15 :10672 MOV WR[0] TO LS[T7.SUB] ;INDICATES NO R80s
U OAC3, 3E0E,15 :10673 MOV LS[PRINT.R80] TO WR[0] ;SET UP TO PRINT NO R80s
U OAC4, 364E,15 :10674 MOV WR[0] TO LS[CONTROL.STATUS]
U OAC5, 3E80,15 :10675 MISC [SET.CP.ATTN] ;ISSUE CPU ATTN TO CONSOLE
:10676 WAIT.TE.2: ; TO GO PRINT MESSAGE
U OAC7, 08AC,74 :10677 JMP [WAIT.TE.2] ;LOOP HERE WAITING FOR
:10678 ; CONSOLE TO RESPOND
:10679
:10680 SAVE.INFO.TE.1:
:10681
U OAC8, 3690,15 :10682 MOV LS[ERPOR.CONTROL] TO WR[0] ;IS APT PRESENT?
:10683 BIT LS[APT.PRESENT] WITH WR[0],
U OAC9, 594A,35 :10684 DT(LONG)&SET.ALU.CC
U OACA, 08AD,D9 :10685 JMP.IF[BITS.CLR] TO [R80.CHK.TE] ;JMP IF NOT UNDER APT
:10686
U OACB, 3646,95 :10687 MOV LS[#8] TO WR[1] ;USE WR[1] AS A COUNTER
U OACC, B692,15 :10688 MOV LS[APT.PARAM] TO WR[0] ;GET THE APT DISK DRIVE CONFIG
:10689 SHFT.TO.BYTE0.TE:
U OACD, A350,15 :10690 ASHR WR[0] ;SHFT THE INFO INTO BYTE 0
:10691 DEC WR[1],
U OACE, A102,B5 :10692 DT(LONG)&SET.ALU.CC
:10693
U OACF, 08AC,D1 :10694 JMP.IF[NEQ] TO [SHFT.TO.BYTE0.TE] ;JMP UNTIL DONE
:10695
U OAD0, 362C,95 :10696 MOV LS[FFFFFFF00] TO WR[1] ;MASK ALL EXCEPT BYTE 0
U OAD1, AF42,15 :10697 BIC WR[1] TO WR[0] ; OF THE APT INFO
:10698
U OAD2, A001,95 :10699 MOV WR[0] TO WR[3] ;SAVE INFO IN WR[3]
:10700
:10701 CMP LS[T9] WITH WR[0],
U OAD3, CE12,35 :10702 DT(LONG)&SET.ALU.CC ;CMP THE CONFIG FOUND BY THIS
:10703 ; TEST AND THE APT CONFIG
U OAD4, 88AE,99 :10704 JMP.IF[EQL] TO [APT.OK.TE] ;JMP IF SAME TO VALIDATE
:10705 ; LS[DRIVE.STATUS]
:10706

```

[illegible]



|              |   |           |        |        |                    |               |                    |                |           |      |
|--------------|---|-----------|--------|--------|--------------------|---------------|--------------------|----------------|-----------|------|
| ZZ-ENKCF-1.4 | : | ENKCF.MIC | Test F | FIFO A | ADRS15-MAR-83      | Fiche 2       | Frame 13           | Sequence 240   |           | ZZ-1 |
| :            | : | ENKCF.MCR | MICRO2 | 1M(01) | 15-MAR-83 11:46:22 | ENKCF         | Micro-diagnostic   | for IDC module | Rev 01.40 | :    |
| :            | : | ENKCF.MIC | Test F | FIFO A | ADRS CNTR MAX L    | BRANCH TEST 1 | (M8388 IDC MODULE) |                | Page 234  | :    |

  

|       |   |                   |                                                                          |        |                 |               |                     |  |  |      |
|-------|---|-------------------|--------------------------------------------------------------------------|--------|-----------------|---------------|---------------------|--|--|------|
| 10761 | : | PAGE              | 'Test F                                                                  | FIFO A | ADRS CNTR MAX L | BRANCH TEST 1 | (M8388 IDC MODULE)' |  |  |      |
| 10762 | : | ++                |                                                                          |        |                 |               |                     |  |  | U 01 |
| 10763 | : |                   |                                                                          |        |                 |               |                     |  |  | U 01 |
| 10764 | : | Test Description: |                                                                          |        |                 |               |                     |  |  | U 01 |
| 10765 | : |                   |                                                                          |        |                 |               |                     |  |  |      |
| 10766 | : |                   | This test is designed to insure that MAX L asserts on the correct        |        |                 |               |                     |  |  |      |
| 10767 | : |                   | FIFO A address count (FF HEX) for an RL02 disk drive.                    |        |                 |               |                     |  |  | U 01 |
| 10768 | : |                   | This test will also verify that when a CLEAR.FIFO.CNTR is issued,        |        |                 |               |                     |  |  |      |
| 10769 | : |                   | the FIFO address lines all clear.                                        |        |                 |               |                     |  |  |      |
| 10770 | : |                   |                                                                          |        |                 |               |                     |  |  |      |
| 10771 | : | Assumptions:      |                                                                          |        |                 |               |                     |  |  | U 01 |
| 10772 | : |                   |                                                                          |        |                 |               |                     |  |  |      |
| 10773 | : |                   | It is assumed that XFER.REQ works correctly.                             |        |                 |               |                     |  |  | U 0  |
| 10774 | : |                   |                                                                          |        |                 |               |                     |  |  | U 0  |
| 10775 | : | Test Steps:       |                                                                          |        |                 |               |                     |  |  |      |
| 10776 | : |                   |                                                                          |        |                 |               |                     |  |  | U 0  |
| 10777 | : |                   | 1. Set up error mask, error number and module number in LS               |        |                 |               |                     |  |  | U 0  |
| 10778 | : |                   | (for error printouts) and clear the previous error number in LS.         |        |                 |               |                     |  |  |      |
| 10779 | : |                   | 2. If the DRIVE SIZING TEST has not been run previously then go run the  |        |                 |               |                     |  |  | U 0  |
| 10780 | : |                   | DISK DRIVE SIZING ROUTINE.                                               |        |                 |               |                     |  |  |      |
| 10781 | : |                   | 3. Force the IDC microcode to DIAG.IDLE.                                 |        |                 |               |                     |  |  | U 0  |
| 10782 | : |                   | 4. PSEL FIFO A.                                                          |        |                 |               |                     |  |  | U 0  |
| 10783 | : |                   | 5. Set up the CSR to branch to the correct IDC microcode routine.        |        |                 |               |                     |  |  |      |
| 10784 | : |                   |                                                                          |        |                 |               |                     |  |  |      |
| 10785 | : |                   | *****                                                                    |        |                 |               |                     |  |  | U 0  |
| 10786 | : |                   | * IDC *                                                                  |        |                 |               |                     |  |  | U 0  |
| 10787 | : |                   | *****                                                                    |        |                 |               |                     |  |  |      |
| 10788 | : |                   |                                                                          |        |                 |               |                     |  |  |      |
| 10789 | : |                   | IDC1. Select FIFO A, UCM = 5.                                            |        |                 |               |                     |  |  | U 0  |
| 10790 | : |                   |                                                                          |        |                 |               |                     |  |  | U 0  |
| 10791 | : |                   | 6. Clear the FIFO ADRS CNTR.                                             |        |                 |               |                     |  |  | U 0  |
| 10792 | : |                   | 7. Set up the CSR to branch to the correct IDC microcode loop.           |        |                 |               |                     |  |  |      |
| 10793 | : |                   |                                                                          |        |                 |               |                     |  |  |      |
| 10794 | : |                   | *****                                                                    |        |                 |               |                     |  |  |      |
| 10795 | : |                   | * IDC *                                                                  |        |                 |               |                     |  |  | U 0  |
| 10796 | : |                   | *****                                                                    |        |                 |               |                     |  |  | U 0  |
| 10797 | : |                   | The IDC microcode will continue looping on the following                 |        |                 |               |                     |  |  |      |
| 10798 | : |                   | steps until MAX L asserts or a CLEAR IDC is issued.                      |        |                 |               |                     |  |  |      |
| 10799 | : |                   |                                                                          |        |                 |               |                     |  |  |      |
| 10800 | : |                   | IDC2. Select BEN1/MAX L.                                                 |        |                 |               |                     |  |  | U 0  |
| 10801 | : |                   | IDC3. If MAX L is asserted then assert XFER REQ and go to                |        |                 |               |                     |  |  | U 0  |
| 10802 | : |                   | DIAG.WAIT.                                                               |        |                 |               |                     |  |  |      |
| 10803 | : |                   | IDC4. Go to IDC2.                                                        |        |                 |               |                     |  |  |      |
| 10804 | : |                   |                                                                          |        |                 |               |                     |  |  |      |
| 10805 | : |                   | 8. Check that XFER REQ is unasserted.(XFER REQ will assert if MAX L is   |        |                 |               |                     |  |  | U 0  |
| 10806 | : |                   | asserted)                                                                |        |                 |               |                     |  |  |      |
| 10807 | : |                   | 9. Increment the error number.                                           |        |                 |               |                     |  |  | U 0  |
| 10808 | : |                   | 10. Select an RL02 drive or a line with no drive attached.               |        |                 |               |                     |  |  |      |
| 10809 | : |                   | 11. Set WR[3] = 0 to use as a counter.                                   |        |                 |               |                     |  |  |      |
| 10810 | : |                   | 12. Write a byte of data to FIFO A to increment the FIFO ADRS CNTR by 1. |        |                 |               |                     |  |  | U 0  |
| 10811 | : |                   | 13. Increment WR[3].                                                     |        |                 |               |                     |  |  |      |
| 10812 | : |                   | 14. Check that XFER REQ is unasserted.                                   |        |                 |               |                     |  |  | U 0  |
| 10813 | : |                   | 15. Repeat Steps 12 - 15 until WR[3] = 254 (FE HEX).                     |        |                 |               |                     |  |  | U 0  |
| 10814 | : |                   | 16. Increment the error number.                                          |        |                 |               |                     |  |  | U 0  |
| 10815 | : |                   | 17. Write a byte of data to FIFO A to increment the FIFO ADRS CNTR by 1. |        |                 |               |                     |  |  | U 0  |

10816  
10817  
10818  
10819  
10820  
10821  
10822  
10823  
10824  
10825  
10826  
10827  
10828  
10829  
10830  
10831  
10832  
10833  
10834  
10835  
10836  
10837  
10838  
10839  
10840  
10841  
10842  
10843  
10844  
10845  
10846  
10847  
10848  
10849  
10850  
10851  
10852  
10853  
10854  
10855  
10856  
10857  
10858  
10859  
10860  
10861  
10862  
10863  
10864  
10865  
10866  
10867  
10868  
10869  
10870

18. Increment WR[3].  
19. Check that XFER REQ is asserted.(XFER REQ asserts if MAX L asserted)  
20. CLEAR IDC.  
21. End test.

Errors:

Printout:

|          |          |            |
|----------|----------|------------|
| EXPECTED | RECEIVED | OTHER DATA |
| N/A      | N/A      | WR[3]      |
|          |          | LOOP CNT   |

ERROR 1 - PCLR FIFO CNTR failed or BEN1/FIFO MAX L failed unasserted.

ERROR 2 - PINCR FIFO CNTR failed or MAX L asserted too soon for an RL02 drive.

ERROR 3 - MAX L did not assert at 255 for an RL02 drive.

Debug:

A disk drive that will assert L RL02 H at the CSR SELECTED  
STAUS LATCH PAL will be selected.  
L RL02 H is inverted to become R80 H = L.

ERROR 1:

The CPU will issue PORT.CONTROL [CLEAR.FIFO.CNTR] which will  
be decoded by the PORT INTERFACE REG PAL.  
The inputs to the PORT INTERFACE REG PAL will be as follows:

CSR 17 H = H  
CSR 14 H = H  
CSR 12 H = L  
CSR 11 H = L  
CSR 10 H = L  
PORT INSTR H = H  
CPU P2 H = H

When CPU CLOCK H clocks the PAL, PCLR FIFO CNTR H will assert. This  
signal is an input to the FIFO ADRS CNTR PALS and will clear all of  
the address lines and deassert FIFO MAX L.

When the BEN1/FIFO.MAX.L IDC microinstruction is executed:

BEN1 S0 H = H  
BEN1 S1 H = L  
BEN1 S2 H = H

This will select FIFO MAX L as NUA 1 H. The IDC microcode will  
remain in the loop waiting for FIFO MAX L to assert and will  
not assert XFER REQ H.  
The CPU will then check to insure that XFER REQ H did not  
assert by executing a SKIP.IF[NO.FAST.INTERRUPT].

```

:10871 : ERROR 2:
:10872 :
:10873 : FIFO A is selected by the port by a PORT.CONTROL [SEL.FIFO.A]
:10874 : instruction so that the FIFO CNTR can be incremented from the
:10875 : PORT. This instruction is decoded by the PORT INTERFACE REG PAL.
:10876 : The inputs will be:
:10877 : CSR 17 H = H
:10878 : CSR 14 H = H
:10879 : CSR 12 H = H
:10880 : CSR 11 H = H
:10881 : CSR 10 H = L
:10882 : PORT INSTR H = H
:10883 : CPU P2 H = H
:10884 : When the PAL is clocked by CPU CLOCK H, PSEL FIFO A H will assert.
:10885 :
:10886 : FIFO A will also be selected by the IDC microcode to allow FIFO A MAX L
:10887 : to be asserted. The IDC microcode will issue UCMD/SET.FIFOA. This will
:10888 : assert UCMD 2 H and UCMD 0 H of the IDC microword. These bits are
:10889 : decoded by the CONTROL STATUS REG PAL. The inputs will be:
:10890 : UCMD 2 H = H
:10891 : UCMD 1 H = L
:10892 : UCMD 0 H = H
:10893 : When the PAL is clocked by P2 CLOCK L, USEL FIFOB H will deassert and
:10894 : USEL FIFOA H being the inverse of USEL FIFOB H, will assert.
:10895 :
:10896 : A PORT.WRITE PORT.ADRS[DATA.BYTE] is issued from the CPU. This
:10897 : instruction will be decoded by the PORT RD/WR DECODE PAL.
:10898 : The inputs to this PAL should be:
:10899 : CSR 17 H = H
:10900 : CSR 14 H = L
:10901 : CSR 13 H = H
:10902 : CSR 12 H = L
:10903 : CSR 11 H = H
:10904 : CSR 10 H = L
:10905 : PORT INSTR H = H
:10906 : CPU P2 H = H
:10907 : This combination of inputs should assert BYTE L and WRITE DATA L, and
:10908 : STROBE DATA H.
:10909 : BYTE L and WRITE DATA L are inputs to the PORT USEQ A & B PALS and are
:10910 : clocked by CPU CLOCK H. This combination of inputs will assert
:10911 : PINCR FIFO CNTR H. This signal is ANDED with PSEL FIFO A H = H and
:10912 : PORT CLOCK L as the clock input to the FIFO A ADRS CNTR HI
:10913 : and LO PALS. When these PALS are clocked, FIFOA ADRS <8:0> H will
:10914 : increment.
:10915 : R80 H will be = L so FIFO MAX L will not assert until
:10916 : FIFOA ADRS<7:4> = H and FIFOA LO FULL H = H and USEL FIFO A H = H.
:10917 :
:10918 : When the BEN1/FIFO.MAX.L IDC microinstruction is executed:
:10919 : BEN1 S0 H = H
:10920 : BEN1 S1 H = L
:10921 : BEN1 S2 H = H
:10922 :
:10923 : This will select FIFO MAX L as NUA 1 H. The IDC microcode will
:10924 : remain in the loop waiting for FIFO MAX L to assert and will
:10925 : not assert XFER REQ H.

```

```

U 0E
U 0E
U 0E

U 0E
U 0E
U 0E

U 0E
U 0E

U 0
U 0
U 0
U 0
U 0
U 0

```

U OE  
U OE  
U OE  
U OE  
U OE  
  
U OE  
U OE  
  
U OE  
U OE  
  
U OE  
  
U O  
U O  
  
U OE  
U OE  
  
U OE

|   |    |
|---|----|
| U | OB |
| U | OB |
| U | OB |
| U | OB |
| U | OB |
| U | OB |
|   |    |
| U | OE |
| U | OE |
|   |    |
| U | OE |
| U | OE |
| U | OE |
|   |    |
| U | OE |
| U | OE |
|   |    |
| U | OE |
| U | OE |
|   |    |
| U | OE |
| U | OE |
| U | OE |
| U | OE |
| U | OE |
|   |    |
| U | OE |
| U | OE |
| U | OE |
| U | OE |
| U | OE |
|   |    |
| U | OE |

```

11036 : =1*1
11037 : TEST.F5:
11038 :
11039 : FUNC/CLR.ONLINE, ;CLR ONLINE FOR THE DRIVE SELECTED
11040 : ; BY DRIVE SEL<1:0> (ONLINE PAL TESTS)
11041 :
11042 : UCMD/SET.FIFOA, ;SELECT FIFO A (USED FOR ANY TEST WHICH
11043 : ; MUST USEL FIFO A
11044 : NAD/XFER.REQ ;DO NOT INCR DAR BETWEEN HERE
11045 : ; AND DIAG.IDLE
11046 : -----
11047 : 1D8:
11048 : XFER.REQ:
11049 :
11050 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
11051 : NAD/DIAG.WAIT
11052 : -----
11053 : 1D7:
11054 : DIAG.WAIT:
11055 :
11056 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
11057 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
11058 :
11059 : -----
11060 : 159:
11061 : =*0*
11062 : DIAG.WAIT1:
11063 :
11064 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
11065 : ; COMMAND FROM THE CPU
11066 : -----
11067 : 15B:
11068 : =*1*
11069 : DIAG.WAIT2:
11070 :
11071 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
11072 : ; WAIT SOME MORE
11073 :
11074 : -----
11075 :
11076 : WAIT.IDC.TF.1:
11077 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ TO SET
11078 : JMP [IDC.DONE.TF.1] ;XFER REQ ASSERTED
11079 : JMP [WAIT.IDC.TF.1] ;WAIT SOME MORE
11080 :
11081 : IDC.DONE.TF.1:
11082 : MOV LS[SETCSR.F0] TO WR[0] ;SET UP TO CLR CSR F<2:0>
11083 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
11084 : MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
11085 :
11086 : LOOP.TF.1:
11087 : MOV LS[#1] TO WR[0]
11088 : MOV WR[0] TO LS[ERROR.NUMBER] ;ERROR 1
11089 :
11090 : PORT.CONTROL [CLEAR.FIFO.CNTR] ;CLEAR FIFO A ADRS CNTR

```

U OB0A, DB00,1A  
 U OB0B, 88B0,D4  
 U OB0C, 08B0,A4  
 U OB0D, B700,15  
 U OB0E, 17A0,15  
 U OB0F, 90FA,15  
 U OB10, B640,15  
 U OB11, BE82,15  
 U OB12, 17C0,15

```

:11091
U OB13, 3712,95 :11092 MOV LS[SETCSR.CRDY] TO WR[1] ;SET UP THE CSR TO BRANCH TO
U OB14, 3720,15 :11093 MOV LS[SETCSR.MAINT] TO WR[0] ; THE CORRECT IDC ROUTINE
U OB15, AEC2,15 :11094 BIS WR[1] TO WR[0] ; THAT WILL LOOP UNTIL MAX L
U OB16, B704,95 :11095 MOV LS[SETCSR.F2] TO WR[1] ; ASSERTS
U OB17, AEC2,15 :11096 BIS WR[1] TO WR[0]
U OB18, AEC4,15 :11097 BIS WR[2] TO WR[0] ;SELECT AN RLO2
U OB19, 17A0,15 :11098 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:11099 ;MAINT=1, CRDY=1, F<2:0>=010
:11100
:11101
:11102 *****
:11103 * IDC *
:11104 *****
:11105
:11106 :018:
:11107 :=000
:11108 :DIAG.IDLE:
:11109
:11110 BEN0/F0,
:11111 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:11112 BEN2/F2,
:11113 NAD/DIAG.IDLE
:11114
:11115 :01A:
:11116 :=010
:11117
:11118 BEN0/CRDY.L,
:11119 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:11120 NAD/TEST.FUNC2 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:11121
:11122 :066:
:11123 :=1*0
:11124 :TEST.FIFO.MAX.1:
:11125 BEN1/FIFO.MAX.L, ;IS FIFO MAX ASSERTED?
:11126 NAD/TEST.FIFO.MAX.2
:11127
:11128 :18D:
:11129 :=0*
:11130 :TEST.FIFO.MAX.2:
:11131 NAD/XFER.REQ ;MAX ASSERTED
:11132
:11133 :18F:
:11134 :=1*
:11135 :TEST.FIFO.MAX.3:
:11136 NAD/TEST.FIFO.MAX.1 ;MAX NOT ASSERTED...LOOP
:11137
:11138
U OB1A, 3792,15 :11139 MOV LS[#6] TO WR[0] ;NOP TO WAIT FOR IDC
:11140 :NOP.TF.1:
:11141 DEC WR[0],
:11142 DT(LONG)&SET.ALU.CC
:11143 JMP.IF[NEQ] TO [NOP.TF.1]
:11144
U OB1D, DB00,1A :11145 SKIP.IF[NO.FAST.INTERRUPT] ;SEE IF MAX L ASSERTED

```

C 4

|              |               |                                                         |                                       |              |          |
|--------------|---------------|---------------------------------------------------------|---------------------------------------|--------------|----------|
| ZZ-ENKCF-1.4 | ENKCF.MIC     | Test F FIFO A ADRS15-MAR-83                             | Fiche 2 Frame C4                      | Sequence 247 |          |
| ENKCF.MCR    | MICRO2 1M(01) | 15-MAR-83 11:46:22                                      | ENKCF Micro-diagnostic for IDC module | Rev 01.40    | Page 241 |
| ENKCF.MIC    | Test F        | FIFO A ADRS CNTR MAX L BRANCH TEST 1 (M8388 IDC MODULE) |                                       |              |          |

  

|                 |        |                                            |                                 |                  |  |
|-----------------|--------|--------------------------------------------|---------------------------------|------------------|--|
| U 0B1E, 88B4.04 | :11146 | JMP [ERROR.TF.1]                           | ;MAX L ASSERTED WHEN FIFO       | ; ADRS WAS CLEAR |  |
|                 | :11147 |                                            |                                 |                  |  |
|                 | :11148 |                                            |                                 |                  |  |
| U 0B1F, 2F82.95 | :11149 | CLR WR[1]                                  | ;CALL CHECK.RESULT TO LOCK      |                  |  |
| U 0B20, 2F80.15 | :11150 | CLR WR[0]                                  | ; IN INTERMITTENT ERRORS        |                  |  |
| U 0B21, BE89.95 | :11151 | MOV WR[3] TO LS[ADDRESS.DATA]              |                                 |                  |  |
| U 0B22, 0869.3C | :11152 | JSR [CHECK.RESULT]                         |                                 |                  |  |
| U 0B23, 88B1.04 | :11153 | JMP [LOOP.TF.1]                            |                                 |                  |  |
|                 | :11154 |                                            |                                 |                  |  |
|                 | :11155 |                                            |                                 |                  |  |
|                 | :11156 |                                            |                                 |                  |  |
| U 0B24, 3642.15 | :11157 | MOV LS[#2] TO WR[0]                        | ;ERROR 2                        |                  |  |
| U 0B25, BE82.15 | :11158 | MOV WR[0] TO LS[ERROR.NUMBER]              |                                 |                  |  |
|                 | :11159 |                                            |                                 |                  |  |
| U 0B26, 378A.15 | :11160 | MOV LS[#FE] TO WR[0]                       | ;ONE BEFORE MAX CNT FOR AN RLO2 |                  |  |
| U 0B27, 3E10.15 | :11161 | MOV WR[0] TO LS[T8]                        |                                 |                  |  |
|                 | :11162 |                                            |                                 |                  |  |
| U 0B28, 2F87.95 | :11163 | CLR WR[3]                                  | ;USE TO TRACK FIFO ADRS         |                  |  |
|                 | :11164 | INCR.CNTR.TF.1:                            |                                 |                  |  |
| U 0B29, 97A8.15 | :11165 | PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] | ;WRITE A BYTE OF UNKNOWN DATA   |                  |  |
|                 | :11166 |                                            | ; TO FIFO A TO INCR ADRS CNTR   |                  |  |
| U 0B2A, 2047.95 | :11167 | INC WR[3]                                  | ;INCR ADRS TRACKER              |                  |  |
| U 0B2B, B788.15 | :11168 | MOV LS[#3] TO WR[0]                        | ;NOP TO WAIT FOR IDC            |                  |  |
|                 | :11169 | NOP.TF.2:                                  |                                 |                  |  |
|                 | :11170 | DEC WR[0]                                  |                                 |                  |  |
| U 0B2C, A100.35 | :11171 | DT(LONG)&SET.ALU.CC                        |                                 |                  |  |
| U 0B2D, 88B2.C1 | :11172 | JMP.IF[NEQ] TO [NOP.TF.2]                  |                                 |                  |  |
|                 | :11173 |                                            |                                 |                  |  |
| U 0B2E, DB00.1A | :11174 | SKIP.IF[NO.FAST.INTERRUPT]                 | ;CHECK IF MAX L ASSERTED        |                  |  |
| U 0B2F, 88B4.04 | :11175 | JMP [ERROR.TF.1]                           | ;MAX L ASSERTED TOO SOON        |                  |  |
|                 | :11176 |                                            |                                 |                  |  |
| U 0B30, 2F82.95 | :11177 | CLR WR[1]                                  | ;CALL CHECK.RESULT TO LOCK      |                  |  |
| U 0B31, 2F80.15 | :11178 | CLR WR[0]                                  | ; IN INTERMITTENT ERRORS        |                  |  |
| U 0B32, BE89.95 | :11179 | MOV WR[3] TO LS[ADDRESS.DATA]              |                                 |                  |  |
| U 0B33, 0869.3C | :11180 | JSR [CHECK.RESULT]                         |                                 |                  |  |
| U 0B34, 88B1.04 | :11181 | JMP [LOOP.TF.1]                            |                                 |                  |  |
|                 | :11182 |                                            |                                 |                  |  |
|                 | :11183 |                                            |                                 |                  |  |
|                 | :11184 |                                            |                                 |                  |  |
|                 | :11185 | CMP LS[T8] WITH WR[3],                     | ;LAST COUNT?                    |                  |  |
| U 0B35, 4E11.B5 | :11186 | DT(LONG)&SET.ALU.CC                        |                                 |                  |  |
| U 0B36, 88B2.91 | :11187 | JMP.IF[NEQ] TO [INCR.CNTR.TF.1]            | ;JUMP IF NOT LAST COUNT         |                  |  |
|                 | :11188 |                                            |                                 |                  |  |
| U 0B37, B788.15 | :11189 | MOV LS[#3] TO WR[0]                        | ;SET UP FOR ERROR 3             |                  |  |
| U 0B38, BE82.15 | :11190 | MOV WR[0] TO LS[ERROR.NUMBER]              | ;WRITE A BYTE OF DATA TO FIFO A |                  |  |
| U 0B39, 97A8.15 | :11191 | PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] | ; TO INCR ADRS CNTR TO MAX (FF) |                  |  |
|                 | :11192 |                                            | ;INCR THE ADRS TRACKER          |                  |  |
| U 0B3A, 2047.95 | :11193 | INC WR[3]                                  | ;NOP TO WAIT FOR IDC            |                  |  |
| U 0B3B, B788.15 | :11194 | MOV LS[#3] TO WR[0]                        |                                 |                  |  |
|                 | :11195 | NOP.TF.3:                                  |                                 |                  |  |
|                 | :11196 | DEC WR[0]                                  |                                 |                  |  |
| U 0B3C, A100.35 | :11197 | DT(LONG)&SET.ALU.CC                        |                                 |                  |  |
| U 0B3D, 08B3.C1 | :11198 | JMP.IF[NEQ] TO [NOP.TF.3]                  |                                 |                  |  |
|                 | :11199 |                                            |                                 |                  |  |
| U 0B3E, DB00.1A | :11200 | SKIP.IF[NO.FAST.INTERRUPT]                 | ;MAX L ASSERTED?                |                  |  |

ZZ-  
:  
:



```

U OB3F, 88B4,54 ;11201 JMP [CALL.CH.RESULT.TF.1] ;MAX L ASSERTED OK
 ;11202
U OB40, 3640,95 ;11203 ERROR.TF.1:
U OB41, 2F80,15 ;11204 MOV LS[#1] TO WR[1] ;FORCE AN ERROR
U OB42, BE89,95 ;11205 CLR WR[0]
U OB43, 0869,3C ;11206 MOV WR[3] TO LS[ADDRESS.DATA] ;PRINT OTHER DATA
U OB44, 88B1,04 ;11207 JSR [CHECK.RESULT]
 ;11208 JMP [LOOP.TF.1] ;LOOP HERE IF ERR & LOE IS SET
 ;11209 ;ELSE SKIP OTHER ERRORS AND END
 ;11210 ; TEST
U OB45, 2F82,95 ;11211 CALL.CH.RESULT.TF.1:
U OB46, 2F80,15 ;11212 CLR WR[1] ;CALL CHECK.RESULT TO LOCK
U OB47, BE89,95 ;11213 CLR WR[0] ; IN INTERMITTENT ERRORS
U OB48, 0869,3C ;11214 MOV WR[3] TO LS[ADDRESS.DATA]
U OB49, 88B1,04 ;11215 JSR [CHECK.RESULT]
 ;11216 JMP [LOOP.TF.1]
 ;11217
 ;11218 CLEANUP.TF.1:
U OB4A, 8875,3C ;11219 JSR [CLEANUP] ;CLEANUP THE CONDITIONS SET BY
 ;11220 ; THIS TEST
 ;11221 END.TF: ;END TEST
 ;11222
 ;11223
 ;11224
 ;11225

```

[illegible]

```

:11281 : 18. Write a byte of data to FIFO A to increment the FIFO ADRS CNTR by 1.
:11282 : 19. Increment WR[3].
:11283 : 20. Check that XFER REQ is asserted.(XFER REQ asserts if MAX L asserted)
:11284 : 21. CLEAR IDC.
:11285 : 22. End test.

```

```

:11286 :
:11287 : Errors:
:11288 :
:11289 : Printout:
:11290 : EXPECTED RECEIVED OTHER DATA
:11291 : N/A N/A WR[3]
:11292 :
:11293 : ERROR 1 - PCLR FIFO CNTR failed or BEN1/FIFO MAX L failed unasserted.
:11294 :
:11295 : ERROR 2 - PINCR FIFO CNTR failed or MAX L asserted too soon for an
:11296 : R80 drive.
:11297 :
:11298 : ERROR 3 - MAX L did not assert at 511 for an R80 drive.
:11299 :

```

```

:11300 :
:11301 : Debug:
:11302 : An R80 disk drive will be selected which will deassert L RL02 H at
:11303 : the CSR SELECTED STATUS LATCH PAL.
:11304 : L RL02 H = L is inverted to become R80 H = H.

```

```

:11305 : ERROR 1:
:11306 :
:11307 : The CPU will issue PORT.CONTROL [CLEAR.FIFO.CNTR] which will
:11308 : be decoded by the PORT INTERFACE REG PAL.
:11309 : The inputs to the PORT INTERFACE REG PAL will be as follows:

```

```

:11310 :
:11311 : CSR 17 H = H
:11312 : CSR 14 H = H
:11313 : CSR 12 H = L
:11314 : CSR 11 H = L
:11315 : CSR 10 H = L
:11316 : PORT INSTR H = H
:11317 : CPU P2 H = H

```

```

:11318 : When CPU CLOCK H clocks the PAL, PCLR FIFO CNTR H will assert. This
:11319 : signal is an input to the FIFO ADRS CNTR PALS and will clear all of
:11320 : the address lines and deassert FIFO MAX L.

```

```

:11321 :
:11322 : When the BEN1/FIFO.MAX.L IDC microinstruction is executed:
:11323 : BEN1 S0 H = H
:11324 : BEN1 S1 H = L
:11325 : BEN1 S2 H = H

```

```

:11326 :
:11327 : This will select FIFO MAX L as NUA 1 H. The IDC microcode will
:11328 : remain in the loop waiting for FIFO MAX L to assert and will
:11329 : not assert XFER REQ H.
:11330 : The CPU will then check to insure that XFER REQ H did not
:11331 : assert by executing a SKIP.IF[NO.FAST.INTERRUPT].

```

```

:11332 :
:11333 : ERROR 2:
:11334 :
:11335 : FIFO A is selected by the port by a PORT.CONTROL [SEL.FIFO.A]

```

:11336 : instruction so that the FIFO CNTR can be incremented from the  
:11337 : PORT. This instruction is decoded by the PORT INTERFACE REG PAL.  
:11338 : The inputs will be:  
:11339 : CSR 17 H = H  
:11340 : CSR 14 H = H  
:11341 : CSR 12 H = H  
:11342 : CSR 11 H = H  
:11343 : CSR 10 H = L  
:11344 : PORT INSTR H = H  
:11345 : CPU P2 H = H  
:11346 : When the PAL is clocked by CPU CLOCK H, PSEL FIFO A H will assert.  
:11347 :  
:11348 : FIFO A will also be selected by the IDC microcode to allow FIFO A MAX L  
:11349 : to be asserted. The IDC microcode will issue UCMD/SET.FIFOA. This will  
:11350 : assert UCMD 2 H and UCMD 0 H of the IDC microword. These bits are  
:11351 : decoded by the CONTROL STATUS REG PAL. The inputs will be:  
:11352 : UCMD 2 H = H  
:11353 : UCMD 1 H = L  
:11354 : UCMD 0 H = H  
:11355 : When the PAL is clocked by P2 CLOCK L, USEL FIFOB H will deassert and  
:11356 : USEL FIFOA H being the inverse of USEL FIFOB H, will assert.  
:11357 :  
:11358 : A PORT.WRITE PORT.ADRS[DATA.BYTE] is issued from the CPU. This  
:11359 : instruction will be decoded by the PORT RD/WR DECODE PAL.  
:11360 : The inputs to this PAL should be:  
:11361 : CSR 17 H = H  
:11362 : CSR 14 H = L  
:11363 : CSR 13 H = H  
:11364 : CSR 12 H = L  
:11365 : CSR 11 H = H  
:11366 : CSR 10 H = L  
:11367 : PORT INSTR H = H  
:11368 : CPU P2 H = H  
:11369 : This combination of inputs should assert BYTE L and WRITE DATA L, and  
:11370 : STROBE DATA H.  
:11371 : BYTE L and WRITE DATA L are inputs to the PORT USEQ A & B PALS and are  
:11372 : clocked by CPU CLOCK H. This combination of inputs will assert  
:11373 : PINCR FIFO CNTR H. This signal is ANDED with PSEL FIFO A H = H and  
:11374 : PORT CLOCK L as the clock input to the FIFO A ADRS CNTR HI  
:11375 : and LO PALS. When these PALS are clocked, FIFOA ADRS <8:0> H will  
:11376 : increment.  
:11377 : R80 H will be = H so FIFO MAX L will not assert until  
:11378 : FIFOA ADRS <8:4> = H and FIFOA LO FULL H = H and USEL FIFO A H = H.  
:11379 :  
:11380 : When the BEN1/FIFO.MAX.L IDC microinstruction is executed:  
:11381 : BEN1 S0 H = H  
:11382 : BEN1 S1 H = L  
:11383 : BEN1 S2 H = H  
:11384 :  
:11385 : This will select FIFO MAX L as NUA 1 H. The IDC microcode will  
:11386 : remain in the loop waiting for FIFO MAX L to assert and will  
:11387 : not assert XFER REQ H.  
:11388 : The CPU will then check to insure that XFER REQ H did not  
:11389 : assert by executing a SKIP.IF[NO.FAST.INTERRUPT].  
:11390 :

U 01  
U 01  
U 01

U 01  
U 0  
U 0

U 01  
U 0  
U 0  
U 0

[illegible]

|              |   |           |         |        |                                                  |                                                 |          |              |          |      |
|--------------|---|-----------|---------|--------|--------------------------------------------------|-------------------------------------------------|----------|--------------|----------|------|
| ZZ-ENKCF-1.4 | : | ENKCF.MIC | Test 10 | FIFO A | ADR15-MAR-83                                     | Fiche 2                                         | Frame 14 | Sequence 253 | Page 247 | ZZ-E |
| :            | : | ENKCF.MCR | MICRO2  | 1M(01) | 15-MAR-83 11:46:22                               | ENKCF Micro-diagnostic for IDC module Rev 01.40 |          |              |          | :    |
| :            | : | ENKCF.MIC | Test 10 | FIFO A | ADRS CNTR MAX L BRANCH TEST 2 (M8388 IDC MODULE) |                                                 |          |              |          | :    |

  

|                 |   |       |                                                           |   |                              |     |
|-----------------|---|-------|-----------------------------------------------------------|---|------------------------------|-----|
| U 0B58, D94E,35 | : | 11446 | BIT LS[BIT7] WITH WR[0],                                  | : | BEEN RUN?                    |     |
| U 0B59, 88B5,B1 | : | 11447 | DT(LONG)&SET.ALU.CC                                       |   |                              |     |
|                 | : | 11448 | JMP.IF[BIT.SET] TO [SIZE.OK.T10.1]                        | : | IF YES THEN GO RUN TEST      |     |
|                 | : | 11449 |                                                           |   |                              |     |
| U 0B5A, 8870,0C | : | 11450 | JSR [DRIVE.SIZE]                                          | : | GO RUN DRIVE SIZING ROUTINE  |     |
|                 | : | 11451 | SIZE.OK.T10.1:                                            |   |                              |     |
|                 | : | 11452 |                                                           |   |                              |     |
| U 0B5B, 8872,EC | : | 11453 | JSR [DIAG.CODE]                                           | : | FORCE THE IDC TO DIAG.IDLE   |     |
|                 | : | 11454 |                                                           |   |                              |     |
|                 | : | 11455 | ;SELECT AN R80 DRIVE IF ONE IS AVAILABLE...ELSE SKIP TEST |   |                              |     |
|                 | : | 11456 |                                                           |   |                              |     |
| U 0B5C, B78E,15 | : | 11457 | MOV LS[DRIVE.STATUS] TO WR[0]                             |   |                              |     |
|                 | : | 11458 |                                                           |   |                              |     |
| U 0B5D, 3790,95 | : | 11459 | MOV LS[FFFFFFFF] TO WR[1]                                 |   |                              |     |
| U 0B5E, AF42,15 | : | 11460 | BIC WR[1] TO WR[0]                                        | : | CHECK BITS <3:0>             |     |
|                 | : | 11461 | CMP LS[ZERO] WITH WR[0],                                  | : | ANY R80S?                    |     |
| U 0B5F, CE9C,35 | : | 11462 | DT(LONG)&SET.ALU.CC                                       |   |                              |     |
| U 0B60, 08BB,69 | : | 11463 | JMP.IF[EQL] TO [CLEANUP.T10.1]                            | : | NO R80, CLEANUP AND END TEST |     |
|                 | : | 11464 |                                                           |   |                              |     |
| U 0B61, B78E,15 | : | 11465 | MOV LS[DRIVE.STATUS] TO WR[0]                             |   |                              |     |
|                 | : | 11466 | BIT LS[BIT0] WITH WR[0],                                  | : | IS DRIVE 0 AN R80?           |     |
| U 0B62, 5940,35 | : | 11467 | DT(LONG)&SET.ALU.CC                                       |   |                              |     |
| U 0B63, 08B6,91 | : | 11468 | JMP.IF[BIT.SET] TO [SEL.DR0.T10.1]                        | : | JMP IF DRV 0 = R80           |     |
|                 | : | 11469 |                                                           |   |                              |     |
|                 | : | 11470 | BIT LS[BIT1] WITH WR[0],                                  | : | IS DRIVE 1 AN R80?           |     |
| U 0B64, D942,35 | : | 11471 | DT(LONG)&SET.ALU.CC                                       |   |                              |     |
| U 0B65, 88B6,B1 | : | 11472 | JMP.IF[BIT.SET] TO [SEL.DR1.T10.1]                        | : | JMP IF DRV 1 = R80           |     |
|                 | : | 11473 |                                                           |   |                              |     |
|                 | : | 11474 | BIT LS[BIT2] WITH WR[0],                                  | : | IS DRIVE 2 AN R80?           |     |
| U 0B66, D944,35 | : | 11475 | DT(LONG)&SET.ALU.CC                                       |   |                              |     |
| U 0B67, 88B6,D1 | : | 11476 | JMP.IF[BIT.SET] TO [SEL.DR2.T10.1]                        | : | JMP IF DRV 2 = R80           |     |
| U 0B68, 08B6,F4 | : | 11477 | JMP [SEL.DR3.T10.1]                                       | : | JMP AND SEL DRV 3            |     |
|                 | : | 11478 |                                                           |   |                              |     |
|                 | : | 11479 |                                                           |   |                              |     |
|                 | : | 11480 | SEL.DR0.T10.1:                                            |   |                              | U 0 |
| U 0B69, 3715,15 | : | 11481 | MOV LS[SETCSR.DS0] TO WR[2]                               | : | SAV DRV SEL = 0              |     |
| U 0B6A, 88B7,04 | : | 11482 | JMP [TEST.T10.1]                                          | : | GO DO TEST                   |     |
|                 | : | 11483 |                                                           |   |                              | U 0 |
|                 | : | 11484 | SEL.DR1.T10.1:                                            |   |                              | U 0 |
| U 0B6B, B717,15 | : | 11485 | MOV LS[SETCSR.DS1] TO WR[2]                               | : | SAV DRV SEL = 1              |     |
| U 0B6C, 88B7,04 | : | 11486 | JMP [TEST.T10.1]                                          | : | GO DO TEST                   |     |
|                 | : | 11487 |                                                           |   |                              | U 0 |
|                 | : | 11488 | SEL.DR2.T10.1:                                            |   |                              | U 0 |
| U 0B6D, 3719,15 | : | 11489 | MOV LS[SETCSR.DS2] TO WR[2]                               | : | SAV DRV SEL = 2              |     |
| U 0B6E, 88B7,04 | : | 11490 | JMP [TEST.T10.1]                                          | : | GO DO TEST                   |     |
|                 | : | 11491 |                                                           |   |                              | U 0 |
|                 | : | 11492 | SEL.DR3.T10.1:                                            |   |                              | U 0 |
| U 0B6F, B71B,15 | : | 11493 | MOV LS[SETCSR.DS3] TO WR[2]                               | : | SAV DRV SEL = 3              |     |
|                 | : | 11494 |                                                           |   |                              |     |
|                 | : | 11495 | TEST.T10.1:                                               |   |                              |     |
| U 0B70, 2F87,95 | : | 11496 | CLR WR[3]                                                 | : | USE AS ADDRESS LOOP COUNTER  |     |
|                 | : | 11497 |                                                           |   |                              | U 0 |
| U 0B71, 17D8,15 | : | 11498 | PORT.CONTROL [SEL.FIFO.A]                                 | : | SEL FIFO A FROM THE PORT     |     |
| U 0B72, 3720,15 | : | 11499 | MOV LS[SETCSR.MAINT] TO WR[0]                             | : | SET UP THE CSR TO BRANCH TO  |     |
| U 0B73, 370A,95 | : | 11500 | MOV LS[SETCSR.F5] TO WR[1]                                | : | THE CORRECT IDC ROUTINE      |     |

[illegible]

```

:11556 : ; COMMAND FROM THE CPU
:11557 :
:11558 :-----
:11559 :15B:
:11560 :=*1*
:11561 :DIAG.WAIT2:
:11562 :
:11563 :NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:11564 : ; WAIT SOME MORE
:11565 :
:11566 :-----
:11567 :WAIT.IDC.T10.1:
U OB76, DB00,1A :11568 :SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ TO SET
U OB77, 8887,94 :11569 :JMP [IDC.DONE.T10.1] ;XFER REQ ASSERTED
U OB78, 8887,64 :11570 :JMP [WAIT.IDC.T10.1] ;WAIT SOME MORE
:11571 :
:11572 :IDC.DONE.T10.1:
U OB79, B700,15 :11573 :MOV LS[SETCSR.F0] TO WR[0] ;SET UP TO CLR CSR F<2:0>
U OB7A, 17A0,15 :11574 :PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
U OB7B, 90FA,15 :11575 :MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
:11576 :
:11577 :LOOP.T10.1:
U OB7C, B640,15 :11578 :MOV LS[#1] TO WR[0]
U OB7D, BE82,15 :11579 :MOV WR[0] TO LS[ERROR.NUMBER] ;ERROR 1
:11580 :
:11581 :PORT.CONTROL [CLEAR.FIFO.CNTR] ;CLEAR FIFO A ADRS CNTR
:11582 :
:11583 :MOV LS[SETCSR.CRDY] TO WR[1] ;SET UP THE CSR TO BRANCH TO
U OB7F, 3712,95 :11584 :MOV LS[SETCSR.MAINT] TO WR[0] ; THE CORRECT IDC ROUTINE
U OB80, 3720,15 :11585 :BIS WR[1] TO WR[0] ; THAT WILL LOOP UNTIL MAX L
U OB81, AEC2,15 :11586 :MOV LS[SETCSR.F2] TO WR[1] ; ASSERTS
U OB82, B704,95 :11587 :BIS WR[1] TO WR[0]
U OB83, AEC2,15 :11588 :BIS WR[2] TO WR[0] ;SELECT AN R80
U OB84, AEC4,15 :11589 :PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
U OB85, 17A0,15 :11590 : ;MAINT=1, CRDY=1, F<2:0>=010
:11591 :-----
:11592 :-----
:11593 :*****
:11594 :* IDC *
:11595 :*****
:11596 :-----
:11597 :018:
:11598 :=000
:11599 :DIAG.IDLE:
:11600 :
:11601 :BEN0/F0,
:11602 :BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:11603 :BEN2/F2,
:11604 :NAD/DIAG.IDLE
:11605 :-----
:11606 :01A:
:11607 :=010
:11608 :
:11609 :BEN0/CRDY.L,
:11610 :BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE

```



```

:11611 : NAD/TEST.FUNC2 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:11612 :-----
:11613 :066:
:11614 :=1*0
:11615 :TEST.FIFO.MAX.1:
:11616 : BEN1/FIFO.MAX.L ;IS FIFO MAX ASSERTED?
:11617 : NAD/TEST.FIFO.MAX.2
:11618 :-----
:11619 :18D:
:11620 :=0*
:11621 :TEST.FIFO.MAX.2:
:11622 : NAD/XFER.REQ ;MAX ASSERTED
:11623 :-----
:11624 :18F:
:11625 :=1*
:11626 :TEST.FIFO.MAX.3:
:11627 : NAD/TEST.FIFO.MAX.1 ;MAX NOT ASSERTED...LOOP
:11628 :-----
:11629 :
U OB86, 3792,15 :11630 :MOV LS[#6] TO WR[0] ;NOP TO WAIT FOR IDC
:11631 :NOP.T10.1:
:11632 : DEC WR[0],
U OB87, A100,35 :11633 : DT(LONG)&SET.ALU.CC
U OB88, 0888,71 :11634 : JMP.IF[NEQ] TO [NOP.T10.1]
:11635 :
U OB89, DB00,1A :11636 : SKIP.IF[NO.FAST.INTERRUPT] ;SEE IF MAX L ASSERTED
U OB8A, 088A,C4 :11637 : JMP [ERROR.T10.1] ;MAX L ASSERTED WHEN FIFO
:11638 : ; ADRS WAS CLEAR
:11639 :
U OB8B, 2F82,95 :11640 : CLR WR[1] ;CALL CHECK.RESULT TO LOCK
U OB8C, 2F80,15 :11641 : CLR WR[0] ; IN INTERMITTENT ERRORS
U OB8D, BE89,95 :11642 : MOV WR[3] TO LS[ADDRESS.DATA] ;
U OB8E, 0869,3C :11643 : JSR [CHECK.RESULT]
U OB8F, 8887,C4 :11644 : JMP [LOOP.T10.1]
:11645 :
:11646 :
U OB90, 3642,15 :11647 : MOV LS[#2] TO WR[0] ;SET UP FOR ERROR 2
U OB91, BE82,15 :11648 : MOV WR[0] TO LS[ERROR.NUMBER]
:11649 :
:11650 :
U OB92, 378C,15 :11651 : MOV LS[#1FE] TO WR[0] ;ONE BEFORE MAX CNT FOR AN R80
U OB93, 3E10,15 :11652 : MOV WR[0] TO LS[T8]
:11653 :
:11654 :
U OB94, 2F87,95 :11654 : CLR WR[3] ;USE TO TRACK FIFO ADRS
:11655 : INCR.CNTR.T10.1:
U OB95, 97A8,15 :11656 : PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;WRITE A BYTE OF UNKNOWN DATA
:11657 : ; TO FIFO A TO INCR ADRS CNTR
U OB96, 2C47,95 :11658 : INC WR[3] ;INCR ADRS TRACKER
U OB97, B788,15 :11659 : MOV LS[#3] TO WR[0] ;NOP TO WAIT FOR IDC
:11660 : NOP.T10.2:
:11661 : DEC WR[0],
U OB98, A100,35 :11662 : DT(LONG)&SET.ALU.CC
U OB99, 8889,81 :11663 : JMP.IF[NEQ] TO [NOP.T10.2]
:11664 :
U OB9A, DB00,1A :11665 : SKIP.IF[NO.FAST.INTERRUPT] ;CHECK IF MAX L ASSERTED

```

|              |   |           |         |        |                 |                                  |                                       |              |          |      |
|--------------|---|-----------|---------|--------|-----------------|----------------------------------|---------------------------------------|--------------|----------|------|
| ZZ-ENKCF-1.4 | : | ENKCF.MIC | Test 10 | FIFO A | ADR15-MAR-83    | Fiche 2                          | Frame M4                              | Sequence 257 | Page 251 | ZZ-E |
| :            | : | ENKCF.MCR | MICRO2  | 1M(01) | 15-MAR-83       | 11:46:22                         | ENKCF Micro-diagnostic for IDC module | Rev 01.40    |          | :    |
| :            | : | ENKCF.MIC | Test 10 | FIFO A | ADRS CNTR MAX L | BRANCH TEST 2 (M8388 IDC MODULE) |                                       |              |          | :    |

  

|                 |   |       |                                            |   |                                |
|-----------------|---|-------|--------------------------------------------|---|--------------------------------|
| U 0B9B, 08BA,C4 | : | 11666 | JMP [ERROR.T10.1]                          | : | MAX L ASSERTED TOO SOON        |
|                 | : | 11667 |                                            |   |                                |
| U 0B9C, 2F82,95 | : | 11668 | CLR WR[1]                                  | : | CALL CHECK.RESULT TO LOCK      |
| U 0B9D, 2F80,15 | : | 11669 | CLR WR[0]                                  | : | IN INTERMITTENT ERRORS         |
| U 0B9E, BE89,95 | : | 11670 | MOV WR[3] TO LS[ADDRESS.DATA]              | : |                                |
| U 0B9F, 0869,3C | : | 11671 | JSR [CHECK.RESULT]                         | : |                                |
| U 0BA0, 8887,C4 | : | 11672 | JMP [LOOP.T10.1]                           | : |                                |
|                 | : | 11673 |                                            |   |                                |
|                 | : | 11674 | CMP LS[T8] WITH WR[3],                     | : | LAST COUNT?                    |
| U 0BA1, 4E11,B5 | : | 11675 | DT(LONG)&SET.ALU.CC                        |   |                                |
| U 0BA2, 08B9,51 | : | 11676 | JMP.IF[NEQ] TO [INCR.CNTR.T10.1]           | : | JUMP IF NOT LAST COUNT         |
|                 | : | 11677 |                                            |   |                                |
| U 0BA3, B788,15 | : | 11678 | MOV LS[#3] TO WR[0]                        | : | SET UP FOR ERROR 3             |
| U 0BA4, BE82,15 | : | 11679 | MOV WR[0] TO LS[ERROR.NUMBER]              | : | WRITE A BYTE OF DATA TO FIFO A |
| U 0BA5, 97A8,15 | : | 11680 | PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] | : | TO INCR ADRS CNTR TO MAX (FF)  |
|                 | : | 11681 |                                            |   |                                |
|                 | : | 11682 |                                            |   |                                |
| U 0BA6, 2047,95 | : | 11683 | INC WR[3]                                  | : | INCR THE ADRS TRACKER          |
|                 | : | 11684 |                                            |   |                                |
| U 0BA7, B788,15 | : | 11685 | MOV LS[#3] TO WR[0]                        | : | NOP TO WAIT FOR IDC            |
|                 | : | 11686 | NOP.T10.3:                                 |   |                                |
|                 | : | 11687 | DEC WR[0],                                 |   |                                |
| U 0BA8, A100,35 | : | 11688 | DT(LONG)&SET.ALU.CC                        |   |                                |
| U 0BA9, 88BA,81 | : | 11689 | JMP.IF[NEQ] TO [NOP.T10.3]                 |   |                                |
|                 | : | 11690 |                                            |   |                                |
|                 | : | 11691 |                                            |   |                                |
| U 0BAA, DB00,1A | : | 11692 | SKIP.IF[NO.FAST.INTERRUPT]                 | : | MAX L ASSERTED?                |
| U 0BAB, 08BB,14 | : | 11693 | JMP [CALL.CH.RESULT.T10.1]                 | : | MAX L ASSERTED OK              |
|                 | : | 11694 | ERROR.T10.1:                               | : | MAX'L DID NOT ASSERT           |
| U 0BAC, 3640,95 | : | 11695 | MOV LS[#1] TO WR[1]                        | : | FORCE AN ERROR                 |
| U 0BAD, 2F80,15 | : | 11696 | CLR WR[0]                                  |   |                                |
| U 0BAE, BE89,95 | : | 11697 | MOV WR[3] TO LS[ADDRESS.DATA]              | : | PRINT OTHER DATA               |
| U 0BAF, 0869,3C | : | 11698 | JSR [CHECK.RESULT]                         |   |                                |
| U 0BB0, 8887,C4 | : | 11699 | JMP [LOOP.T10.1]                           | : | LOOP HERE IF ERR & LOE IS SET  |
|                 | : | 11700 |                                            | : | ELSE SKIP OTHER ERRORS AND END |
|                 | : | 11701 |                                            | : | TEST                           |
|                 | : | 11702 | CALL.CH.RESULT.T10.1:                      |   |                                |
| U 0BB1, 2F82,95 | : | 11703 | CLR WR[1]                                  | : | CALL CHECK.RESULT TO LOCK      |
| U 0BB2, 2F80,15 | : | 11704 | CLR WR[0]                                  | : | IN INTERMITTENT ERRORS         |
| U 0BB3, BE89,95 | : | 11705 | MOV WR[3] TO LS[ADDRESS.DATA]              | : |                                |
| U 0BB4, 0869,3C | : | 11706 | JSR [CHECK.RESULT]                         | : |                                |
| U 0BB5, 8887,C4 | : | 11707 | JMP [LOOP.T10.1]                           | : |                                |
|                 | : | 11708 |                                            |   |                                |
|                 | : | 11709 |                                            |   |                                |
|                 | : | 11710 | CLEANUP.T10.1:                             |   |                                |
| U 0BB6, 8875,3C | : | 11711 | JSR [CLEANUP]                              | : | CLEANUP THE CONDITIONS SET BY  |
|                 | : | 11712 |                                            | : | THIS TEST                      |
|                 | : | 11713 | END.T10:                                   | : | END TEST                       |
|                 | : | 11714 |                                            |   |                                |
|                 | : | 11715 |                                            |   |                                |
|                 | : | 11716 |                                            |   |                                |
|                 | : | 11717 |                                            |   |                                |
|                 | : | 11718 |                                            |   |                                |
|                 | : | 11719 |                                            |   |                                |
|                 | : | 11720 |                                            |   |                                |

:11721 :PAGE 'Test 11 FIFO B ADRS CNTR MAX L BRANCH TEST 1 (M8388 IDC MODULE)'  
:11722 :++  
:11723 :Test Description:  
:11724 :  
:11725 :This test is designed to insure that MAX L asserts on the correct  
:11726 :FIFO B address count (FF HEX) for an RL02 disk drive.  
:11727 :This test will also verify that MAX L clears when the CPU issues a  
:11728 :CLEAR.FIFO.CNTR instruction or when the IDC microcode executes a  
:11729 :UCLR FIFO CNTR.  
:11730 :  
:11731 :Assumptions:  
:11732 :  
:11733 :It is assumed that XFER.REQ works correctly.  
:11734 :  
:11735 :Test Steps:  
:11736 :  
:11737 :1. Set up error mask, error number and module number in LS  
:11738 : (for error printouts) and clear the previous error number in LS.  
:11739 :2. If the DRIVE SIZING TEST has not been run previously then go run the  
:11740 : DISK DRIVE SIZING ROUTINE.  
:11741 :3. Force the IDC microcode to DIAG.IDLE.  
:11742 :4. PSEL FIFO B.  
:11743 :5. Set up the CSR to branch to the correct IDC microcode routine.  
:11744 :  
:11745 :\*\*\*\*\*  
:11746 : \* IDC \*  
:11747 :\*\*\*\*\*  
:11748 :  
:11749 :IDC1. Select FIFO A, UCM = 5.  
:11750 :IDC2. Change FIFO, UCM = 6. (to USEL FIFO B)  
:11751 :  
:11752 :6. Set up the CSR to branch to the correct IDC microcode loop.  
:11753 :  
:11754 :\*\*\*\*\*  
:11755 : \* IDC \*  
:11756 :\*\*\*\*\*  
:11757 :The IDC microcode will continue looping on the following  
:11758 :steps until MAX L asserts or a CLEAR IDC is issued.  
:11759 :  
:11760 :IDC3. Select BEN1/MAX L.  
:11761 :IDC4. If MAX L is asserted then assert XFER REQ and go to  
:11762 : DIAG.WAIT.  
:11763 :IDC5. Go to IDC3.  
:11764 :  
:11765 :7. Clear the FIFO ADRS CNTR.  
:11766 :8. Check that XFER REQ is unasserted.(XFER REQ will assert if MAX L is  
:11767 : asserted)  
:11768 :9. Increment the error number.  
:11769 :10. Select an RL02 drive or a line with no drive attached.  
:11770 :11. Set WR[3] = 0 to use as a counter.  
:11771 :12. Write a byte of data to FIFO B to increment the FIFO ADRS CNTR by 1.  
:11772 :13. Increment WR[3].  
:11773 :14. Check that XFER REQ is unasserted.  
:11774 :15. Repeat Steps 12 - 15 until WR[3] = 254 (FE HEX).  
:11775 :16. Increment the error number.



```

11831 : ERROR 2:
11832 :
11833 : FIFO B is selected by the port by a PORT.CONTROL [SEL.FIFO.B]
11834 : instruction so that the FIFO CNTR can be incremented from the
11835 : PORT. This instruction is decoded by the PORT INTERFACE REG PAL.
11836 : The inputs will be:
11837 : CSR 17 H = H
11838 : CSR 14 H = H
11839 : CSR 12 H = H
11840 : CSR 11 H = H
11841 : CSR 10 H = H
11842 : PORT INSTR H = H
11843 : CPU P2 H = H
11844 :
11845 : When the PAL is clocked by CPU CLOCK H, PSEL FIFO A H will deassert.
11846 : PSEL FIFO B H = H, is the inverse of PSEL FIFO A H = L.
11847 :
11848 : FIFO B will also be selected by the IDC microcode to allow FIFO B MAX L
11849 : to be asserted. The IDC microcode will issue UCMD/SET.FIFOA. This will
11850 : assert UCMD 2 H and UCMD 0 H of the IDC microword. These bits are
11851 : decoded by the CONTROL STATUS REG PAL. The inputs will be:
11852 : UCMD 2 H = H
11853 : UCMD 1 H = L
11854 : UCMD 0 H = H
11855 :
11856 : When the PAL is clocked by P2 CLOCK L, USEL FIFOB H will deassert and
11857 : USEL FIFOA H being the inverse of USEL FIFOB H, will assert.
11858 : The IDC microcode will execute a UCMD/CHG.FIFO. This will assert
11859 : UCMD 2 H and UCMD 1 H of the IDC microword. These bits are decoded by
11860 : the CONTROL STATUS REG PAL. The inputs will be:
11861 : UCMD 2 H = H
11862 : UCMD 1 H = H
11863 : UCMD 0 H = L
11864 :
11865 : When the PAL is clocked by P2 CLOCK L, USEL FIFOA H = L will be
11866 : inverted to USEL FIFOB H = H.
11867 :
11868 : A PORT.WRITE PORT.ADRS[DATA.BYTE] is issued from the CPU. This
11869 : instruction will be decoded by the PORT RD/WR DECODE PAL.
11870 : The inputs to this PAL should be:
11871 : CSR 17 H = H
11872 : CSR 14 H = L
11873 : CSR 13 H = H
11874 : CSR 12 H = L
11875 : CSR 11 H = H
11876 : CSR 10 H = L
11877 : PORT INSTR H = H
11878 : CPU P2 H = H
11879 :
11880 : This combination of inputs should assert BYTE L and WRITE DATA L, and
11881 : STROBE DATA H.
11882 : BYTE L and WRITE DATA L are inputs to the PORT USEQ A & B PALS and are
11883 : clocked by CPU CLOCK H. This combination of inputs will assert
11884 : PINCR FIFO CNTR H. This signal is ANDED with PSEL FIFO B H = 4 and
11885 : PORT CLOCK L as the clock input to the FIFO B ADRS CNTR HI
11886 : and LO PALS. When these PALS are clocked, FIFOB ADRS<7:4> will
11887 : increment.

```

|              |   |           |         |                        |               |                     |                                       |           |     |
|--------------|---|-----------|---------|------------------------|---------------|---------------------|---------------------------------------|-----------|-----|
| ZZ-ENKCF-1.4 | : | ENKCF.MIC | Test 11 | FIFO B ADR15-MAR-83    | riche 2       | Frame D5            | Sequence 261                          | Page 255  | ZZ- |
| :            | : | ENKCF.MCR | MICRO2  | 1M(01)                 | 15-MAR-83     | 11:46:22            | ENKCF Micro-diagnostic for IDC module | Rev 01.40 | :   |
| :            | : | ENKCF.MIC | Test 11 | FIFO B ADRS CNTR MAX L | BRANCH TEST 1 | (M8388 IDC MC: JLE) |                                       |           | :   |

  

|                 |   |                                                                     |                              |
|-----------------|---|---------------------------------------------------------------------|------------------------------|
| 11886           | : | FIFO MAX L will not assert because the inputs to FIFOA ADRS CNTR HI |                              |
| 11887           | : | PAL will be USEL FIFOA H = L and FIFOB MAX L = H.                   |                              |
| 11888           | : | Since R80 H will be = L, FIFOB MAX L at the FIFOB ADRS CNTR HI PAL, | U 0                          |
| 11889           | : | will not assert until FIFOB ADRS<7:4> = H and FIFOB LO FULL H = H   | U 0                          |
| 11890           | : | and USEL FIFO B H = H.                                              |                              |
| 11891           | : |                                                                     |                              |
| 11892           | : | When the BEN1/FIFO.MAX.L IDC microinstruction is executed:          | U 0                          |
| 11893           | : | BEN1 S0 H = H                                                       | U 0                          |
| 11894           | : | BEN1 S1 H = L                                                       |                              |
| 11895           | : | BEN1 S2 H = H                                                       |                              |
| 11896           | : |                                                                     | U 0                          |
| 11897           | : | This will select FIFO MAX L as NUA 1 H. The IDC microcode will      |                              |
| 11898           | : | remain in the loop waiting for FIFO MAX L to assert and will        |                              |
| 11899           | : | not assert XFER REQ H.                                              | U 0                          |
| 11900           | : | The CPU will then check to insure that XFER REQ H did not           |                              |
| 11901           | : | assert by executing a SKIP.IF[NO.FAST.INTERRUPT].                   | U 0                          |
| 11902           | : |                                                                     | U 0                          |
| 11903           | : | ERROR 3:                                                            | U 0                          |
| 11904           | : | FIFOB ADRS <7:0> will be = 11111110. The FIFOB ADRS CNTR PALs will  | U 0                          |
| 11905           | : | be clocked one more time by writing a byte of data to FIFO B. This  | U 0                          |
| 11906           | : | will increment FIFOB ADRS <7:0> = 11111111.                         | U 0                          |
| 11907           | : | The inputs to the FIFOB ADRS CNTR HI PAL will be:                   |                              |
| 11908           | : | R80 H = L                                                           |                              |
| 11909           | : | USEL FIFOB H = H                                                    |                              |
| 11910           | : | FIFOB LO FULL H = H                                                 |                              |
| 11911           | : | FIFOB ADRS <7:4> H = 1111                                           |                              |
| 11912           | : | This combination of inputs will assert FIFOB MAX L.                 |                              |
| 11913           | : |                                                                     |                              |
| 11914           | : | The inputs to the FIFOA ADRS CNTR HI PAL will be:                   |                              |
| 11915           | : | FIFOB MAX L = L                                                     |                              |
| 11916           | : | USEL FIFO A H = L                                                   |                              |
| 11917           | : | This combination of inputs will assert FIFO MAX L.                  |                              |
| 11918           | : |                                                                     |                              |
| 11919           | : | When the BEN1/FIFO.MAX.L IDC microinstruction is executed:          |                              |
| 11920           | : | BEN1 S0 H = H                                                       |                              |
| 11921           | : | BEN1 S1 H = L                                                       |                              |
| 11922           | : | BEN1 S2 H = H                                                       |                              |
| 11923           | : |                                                                     |                              |
| 11924           | : | This will select FIFO MAX L as NUA 1 H. The IDC microcode will      |                              |
| 11925           | : | branch to an instruction that will assert XFER REQ H.               |                              |
| 11926           | : | The CPU will then check to insure that XFER REQ H asserted by       |                              |
| 11927           | : | executing a SKIP.IF[NO.FAST.INTERRUPT].                             |                              |
| 11928           | : |                                                                     |                              |
| 11929           | : | --                                                                  |                              |
| 11930           | : | *****                                                               |                              |
| 11931           | : | T.11:                                                               |                              |
| U 0BB7, B65E,15 | : | MOV LS[BEGIN.TEST] TO WR[0]                                         | ;SET BIT 15 IN WR[0] FOR     |
| U 0BB8, 3E80,15 | : | MOV WR[0] TO LS[CONTROL.STATUS]                                     | ; CONTROL AND STATUS WORD    |
|                 | : |                                                                     | ;SET BIT 15 IN CONTROL AND   |
|                 | : |                                                                     | ; STATUS WORD - BIT 15       |
|                 | : |                                                                     | ; INDICATES START OF TEST TO |
|                 | : |                                                                     | ; CONSOLE                    |
| U 0BB9, 10E0,15 | : | MISC [SET.CP.ATTN]                                                  | ;ISSUE CPU ATTN TO CONSOLE   |
| U 0BBA, 88BB,A4 | : | WAIT.T11.0:                                                         |                              |
|                 | : | JMP [WAIT.T11.0]                                                    | ;LOOP HERE WAITING FOR       |

```

;11941 ; CONSOLE TO RESPOND
;11942
U OB BB, 65A0,15 :11943 CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
;11944 ; IN LS
;11945
U OB BC, B64A,15 :11946 MOV LS[IDC] TO WR[0] ;STORE MODULE CODE IN LS TO
U OB BD, 3E8C,15 :11947 MOV WR[0] TO LS[MODULE.NUM] ; INDICATE IDC MODULE
;11948
;11949
;11950
U OB BE, E58A,15 :11951 CLR LS[ERROR.MASK]
;11952
U OB BF, B670,15 :11953 MOV LS[OTHER.DATA] TO WR[0] ;SET UP TO PRINT OTHER DATA
U OB C0, 3E80,15 :11954 MOV WR[0] TO LS[CONTROL.STATUS] ; IF ERROR
;11955
U OB C1, B66E,15 :11956 MOV LS[NA.EXP.REC] TO WR[0] ;SET UP TO PRINT N/A FOR
U OB C2, 6D80,15 :11957 BIS WR[0] TO LS[CONTROL.STATUS] ; EXPECTED & RECEIVED DATA
;11958
;11959
;11960
U OB C3, B78E,15 :11961 MOV LS[DRIVE.STATUS] TO WR[0] ;HAS THE DRIVE SIZING ROUTINE
;11962 BIT LS[BIT7] WITH WR[0], ; BEEN RUN?
U OB C4, D94E,35 :11963 DT(LONG)&SET.ALU.CC
U OB C5, 88BC,71 :11964 JMP.IF[BIT.SET] TO [SIZE.OK.T11.1] ;IF YES THEN GO RUN TEST
;11965
U OB C6, 8870,0C :11966 JSR [DRIVE.SIZE] ;GO RUN DRIVE SIZING ROUTINE
;11967
;11968 SIZE.OK.T11.1:
;11969
U OB C7, 8872,EC :11970 JSR [DIAG.CODE] ;FORCE THE IDC TO DIAG.IDLE
;11971
;11972 ;SELECT AN RL02 DRIVE
;11973
U OB C8, B78E,15 :11974 MOV LS[DRIVE.STATUS] TO WR[0]
;11975 BIT LS[BIT0] WITH WR[0], ;IS DRIVE 0 AN R80?
U OB C9, 5940,35 :11976 DT(LONG)&SET.ALU.CC
U OB CA, 08BC,D9 :11977 JMP.IF[BIT.CLR] TO [SET.DS0.T11.1] ;JMP IF DRV 0 IS NOT AN R80
U OB CB, B717,15 :11978 MOV LS[SETCSR.DS1] TO WR[2] ;SAVE THE DATA TO SEL DRIVE 1
U OB CC, 888C,E4 :11979 JMP [TEST.T11.1]
;11980 SET.DS0.T11.1:
U OB CD, 3715,15 :11981 MOV LS[SETCSR.DS0] TO WR[2] ;SAVE THE DATA TO SEL DRIVE 0
;11982
;11983 TEST.T11.1:
;11984
U OB CE, 2F87,95 :11985 CLR WR[3] ;USE AS ADDR LOOP CNTR
U OB CF, 97DC,15 :11986 PORT.CONTROL [SEL.FIFO.B] ;SEL FIFO B FROM THE PORT
;11987
U OB D0, 3720,15 :11988 MOV LS[SETCSR.MAINT] TO WR[0] ;SET UP THE CSR TO BRANCH TO
U OB D1, 370A,95 :11989 MOV LS[SETCSR.F5] TO WR[1] ; THE CORRECT IDC ROUTINE
U OB D2, AEC2,15 :11990 BIS WR[1] TO WR[0] ; TO USEL FIFO A
U OB D3, 17A0,15 :11991 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
;11992 ;MAINT=1, CRDY=0, F<2:0>=101
;11993
;11994
;11995

```

\*\*\*\*\*

```
:11996 : * IDC *
:11997 : *
:11998 :-----
:11999 :018:
:12000 :=000
:12001 :DIAG.IDLE:
:12002 :
:12003 : BEN0/F0,
:12004 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:12005 : BEN2/F2,
:12006 : NAD/DIAG.IDLE
:12007 :-----
:12008 :01D:
:12009 :=101
:12010 :
:12011 : BEN0/CRDY.L,
:12012 : BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:12013 : NAD/TEST.FUNC5 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:12014 :-----
:12015 :075:
:12016 :=1*1
:12017 :TEST.F5:
:12018 :
:12019 : FUNC/CLR.ONLINE, ;CLR ONLINE FOR THE DRIVE SELECTED
:12020 : ; BY DRIVE SEL<1:0> (ONLINE PAL TESTS)
:12021 :
:12022 : UCMD/SET.FIFOA, ;SELECT FIFO A (USED FOR ANT TEST WHICH
:12023 : ; MUST USEL FIFO A)
:12024 : NAD/XFER.REQ ;DO NOT INCR DAR BETWEEN HERE
:12025 : ; AND DIAG.IDLE
:12026 :-----
:12027 :1D8:
:12028 :XFER.REQ:
:12029 :
:12030 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:12031 : NAD/DIAG.WAIT
:12032 :-----
:12033 :1D7:
:12034 :DIAG.WAIT:
:12035 :
:12036 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:12037 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:12038 :
:12039 :-----
:12040 :159:
:12041 :=*0*
:12042 :DIAG.WAIT1:
:12043 :
:12044 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:12045 : ; COMMAND FROM THE CPU
:12046 :-----
:12047 :
:12048 :158:
:12049 :=*1*
:12050 :DIAG.WAIT2:
```

U  
U  
U  
U  
U  
U  
C  
C  
C  
C  
C  
C







```

:12161 : =000
:12162 : DIAG.IDLE:
:12163 :
:12164 : BEN0/F0,
:12165 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:12166 : BEN2/F2,
:12167 : NAD/DIAG.IDLE
:12168 : -----
:12169 : 01A:
:12170 : =010
:12171 :
:12172 : BEN0/CRDY.L,
:12173 : BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:12174 : NAD/TEST.FUNC2 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:12175 : -----
:12176 : -----
:12177 : 066:
:12178 : =1*0
:12179 : TEST.FIFO.MAX.1:
:12180 : BEN1/FIFO.MAX.L, ;IS FIFO MAX ASSERTED?
:12181 : NAD/TEST.FIFO.MAX.2
:12182 : -----
:12183 : 18D:
:12184 : =0*
:12185 : TEST.FIFO.MAX.2:
:12186 : NAD/XFER.REQ ;MAX ASSERTED
:12187 : -----
:12188 : 18F:
:12189 : =1*
:12190 : TEST.FIFO.MAX.3:
:12191 : NAD/TEST.FIFO.MAX.1 ;MAX NOT ASSERTED...LOOP
:12192 : -----
:12193 :
:12194 : MOV LS[#6] TO WR[0] ;NOP TO WAIT FOR IDC
:12195 : NOP.T11.1:
:12196 : DEC WR[0],
:12197 : DT(LONG)&SET.ALU.CC
:12198 : JMP.IF[NEQ] TO [NOP.T11.1]
:12199 :
:12200 : SKIP.IF[NO.FAST.INTERRUPT] ;SEE IF MAX L ASSERTED
:12201 : JMP [ERROR.T11.1] ;MAX L ASSERTED WHEN FIFO
:12202 : ; ADRS WAS CLEAR
:12203 : ;CALL CHECK.RESULT TO LOCK
:12204 : ; IN INTERMITTENT ERRORS
:12205 :
:12206 : CLR WR[1]
:12207 : CLR WR[0]
:12208 : MOV WR[3] TO LS[ADDRESS.DATA]
:12209 : JSR [CHECK.RESULT]
:12210 : JMP [LOOP.T11.1]
:12211 :
:12212 : MOV LS[#2] TO WR[0] ;SET UP FOR ERROR 2
:12213 : MOV WR[0] TO LS[ERROR.NUMBER]
:12214 :
:12215 : MOV LS[#FE] TO WR[0] ;ONE BEFORE MAX CNT FOR AN RLO2

```

U OBEE, 3792,15

U OBEF, A100,35

U OBF0, 88BE,F1

U OBF1, DB00,1A

U OBF2, 88C1,44

U OBF3, 2F82,95

U OBF4, 2F80,15

U OBF5, BE89,95

U OBF6, 0869,3C

U OBF7, 08BE,44

U OBF8, 3642,15

U OBF9, BE82,15

U OBFA, 378A,15

ZZ-E

:

:

U 00

U 00

U 00

U 00

U 00

U 00

U 00

U 00

U 00

U 00

U 00

U 00

U 00

U 00

U 00

U 00

U 00

U 00

J 5

|              |           |                                                                 |                                       |              |          |
|--------------|-----------|-----------------------------------------------------------------|---------------------------------------|--------------|----------|
| ZZ-ENKCF-1.4 | ENKCF.MIC | Test 11 FIFO B ADR15-MAR-83                                     | Fiche 2 Frame J5                      | Sequence 267 |          |
| :            | ENKCF.MCR | MICRO2 1M(01) 15-MAR-83 11:46:22                                | ENKCF Micro-diagnostic for IDC module | Rev 01.40    | Page 261 |
| :            | ENKCF.MIC | Test 11 FIFO B ADRS CNTR MAX L BRANCH TEST 1 (M8388 IDC MODULE) |                                       |              |          |

  

|                 |        |                                            |  |                                 |  |
|-----------------|--------|--------------------------------------------|--|---------------------------------|--|
| U 0BFB, 3E10,15 | :12216 | MOV WR[0] TO LS[T8]                        |  |                                 |  |
|                 | :12217 |                                            |  |                                 |  |
| U 0BFC, 2F87,95 | :12218 | CLR WR[3]                                  |  | ;USE TO TRACK FIFO ADRS         |  |
|                 | :12219 | INCR.CNTR.T11.1:                           |  |                                 |  |
| U 0BFD, 97A8,15 | :12220 | PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] |  | ;WRITE A BYTE OF UNKNOWN DATA   |  |
|                 | :12221 |                                            |  | ; TO FIFO A TO INCR ADRS CNTR   |  |
| U 0BFE, 2047,95 | :12222 | INC WR[3]                                  |  | ;INCR ADRS TRACKER              |  |
| U 0BFF, B788,15 | :12223 | MOV LS[#3] TO WR[0]                        |  | ;NOP TO WAIT FOR IDC            |  |
|                 | :12224 | NOP.T11.2:                                 |  |                                 |  |
|                 | :12225 | DEC WR[0],                                 |  |                                 |  |
| U 0C00, A100,35 | :12226 | DT(LONG)&SET.ALU.CC                        |  |                                 |  |
| U 0C01, 88C0,01 | :12227 | JMP.IF[NEQ] TO [NOP.T11.2]                 |  |                                 |  |
|                 | :12228 |                                            |  |                                 |  |
| U 0C02, DB00,1A | :12229 | SKIP.IF[NO.FAST.INTERRUPT]                 |  | ;CHECK IF MAX L ASSERTED        |  |
| U 0C03, 88C1,44 | :12230 | JMP [ERROR.T11.1]                          |  | ;MAX L ASSERTED TOO SOON        |  |
|                 | :12231 |                                            |  |                                 |  |
| U 0C04, 2F82,95 | :12232 | CLR WR[1]                                  |  | ;CALL CHECK.RESULT TO LOCK      |  |
| U 0C05, 2F80,15 | :12233 | CLR WR[0]                                  |  | ; IN INTERMITTENT ERRORS        |  |
| U 0C06, BE89,95 | :12234 | MOV WR[3] TO LS[ADDRESS.DATA]              |  |                                 |  |
| U 0C07, 0869,3C | :12235 | JSR [CHECK.RESULT]                         |  |                                 |  |
| U 0C08, 08BE,44 | :12236 | JMP [LOOP.T11.1]                           |  |                                 |  |
|                 | :12237 |                                            |  |                                 |  |
|                 | :12238 |                                            |  |                                 |  |
|                 | :12239 | CMP LS[T8] WITH WR[3],                     |  | ;LAST COUNT?                    |  |
| U 0C09, 4E11,B5 | :12240 | DT(LONG)&SET.ALU.CC                        |  |                                 |  |
| U 0C0A, 88BF,D1 | :12241 | JMP.IF[NEQ] TO [INCR.CNTR.T11.1]           |  | ;JUMP IF NOT LAST COUNT         |  |
|                 | :12242 |                                            |  |                                 |  |
| U 0C0B, B788,15 | :12243 | MOV LS[#3] TO WR[0]                        |  |                                 |  |
| U 0C0C, BE82,15 | :12244 | MOV WR[0] TO LS[ERROR.NUMBER]              |  | ;SET UP FOR ERROR 3             |  |
| U 0C0D, 97A8,15 | :12245 | PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] |  | ;WRITE A BYTE OF DATA TO FIFO A |  |
|                 | :12246 |                                            |  | ; TO INCR ADRS CNTR TO MAX (FF) |  |
| U 0C0E, 2047,95 | :12247 | INC WR[3]                                  |  | ;INCR THE ADRS TRACKER          |  |
|                 | :12248 |                                            |  |                                 |  |
| U 0C0F, B788,15 | :12249 | MOV LS[#3] TO WR[0]                        |  | ;NOP TO WAIT FOR IDC            |  |
|                 | :12250 | NOP.T11.3:                                 |  |                                 |  |
|                 | :12251 | DEC WR[0],                                 |  |                                 |  |
| U 0C10, A100,35 | :12252 | DT(LONG)&SET.ALU.CC                        |  |                                 |  |
| U 0C11, 08C1,01 | :12253 | JMP.IF[NEQ] TO [NOP.T11.3]                 |  |                                 |  |
|                 | :12254 |                                            |  |                                 |  |
|                 | :12255 |                                            |  |                                 |  |
| U 0C12, DB00,1A | :12256 | SKIP.IF[NO.FAST.INTERRUPT]                 |  | ;MAX L ASSERTED?                |  |
| U 0C13, 08C1,94 | :12257 | JMP [CALL.CH.RESULT.T11.1]                 |  | ;MAX L ASSERTED OK              |  |
|                 | :12258 | ERROR.T11.1:                               |  |                                 |  |
| U 0C14, 3640,95 | :12259 | MOV LS[#1] TO WR[1]                        |  | ;FORCE AN ERROR                 |  |
| U 0C15, 2F80,15 | :12260 | CLR WR[0]                                  |  |                                 |  |
| U 0C16, BE89,95 | :12261 | MOV WR[3] TO LS[ADDRESS.DATA]              |  | ;PRINT OTHER DATA               |  |
| U 0C17, 0869,3C | :12262 | JSR [CHECK.RESULT]                         |  |                                 |  |
| U 0C18, 08BE,44 | :12263 | JMP [LOOP.T11.1]                           |  | ;LOOP HERE IF ERR & LOE IS SET  |  |
|                 | :12264 |                                            |  | ;ELSE SKIP OTHER ERRORS AND END |  |
|                 | :12265 |                                            |  | ; TEST                          |  |
|                 | :12266 |                                            |  |                                 |  |
|                 | :12267 | CALL.CH.RESULT.T11.1:                      |  |                                 |  |
| U 0C19, 2F82,95 | :12268 | CLR WR[1]                                  |  | ;CALL CHECK.RESULT TO LOCK      |  |
| U 0C1A, 2F80,15 | :12269 | CLR WR[0]                                  |  | ; IN INTERMITTENT ERRORS        |  |
| U 0C1B, BE89,95 | :12270 | MOV WR[3] TO LS[ADDRESS.DATA]              |  |                                 |  |

ZZ-E  
:  
:

ZZ-ENKCF-1.4 : ENKCF.MIC Test 11 FIFO B ADR15-MAR-83 K 5 Fiche 2 Frame K5 Sequence 268  
 : ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40 Page 262  
 : ENKCF.MIC Test 11 FIFO B ADRS CNTR MAX L BRANCH TEST 1 (M8388 IDC MODULE)

U OC1C, 0869,3C :12271 JSR [CHECK.RESULT]  
 U OC1D, 08BE,44 :12272 JMP [LOOP.T11.1]

:12273  
 :12274  
 :12275 CLEANUP.T11.1:  
 U OC1E, 8875,3C :12276 JSR [CLEANUP]  
 :12277  
 :12278 END.T11:  
 :12279  
 :12280  
 :12281

;CLEANUP THE CONDITIONS SET BY  
 ; THIS TEST  
 ;END TEST

ZZ-E  
 :  
 :

```
:12282 .PAGE 'Test 12 FIFO B ADRS CNTR MAX L BRANCH TEST 2 (M8388 IDC MODULE)''
:12283 .++
:12284
:12285 .Test Description:
:12286
:12287 .This test will only be run if an R80 is available on the system.
:12288 .This test is designed to verify that MAX L asserts on the correct
:12289 .address count (1FF HEX) of FIFO B when an R80 is selected.
:12290
:12291 .Assumptions:
:12292
:12293 .It is assumed that XFER.REQ works correctly.
:12294
:12295 .Test Steps:
:12296
:12297 .1. Set up error mask, error number and module number in LS
:12298 .(for error printouts) and clear the previous error number in LS.
:12299 .2. If the DRIVE SIZING TEST has not been run previously then
:12300 .go run the DISK DRIVE SIZING ROUTINE.
:12301 .3. If an R80 is not available then print 'R80 NOT PRESENT'
:12302 .and go to end of test.
:12303 .4. Force the IDC microcode to DIAG.IDLE.
:12304 .5. PSEL FIFO B.
:12305 .6. Set up the CSR to branch to the correct IDC microcode routine.
:12306
:12307 .*****
:12308 .* IDC *
:12309 .*****
:12310
:12311 .IDC1. Select FIFO A, UCM = 5.
:12312 .IDC2. Change FIFO, UCM = 6. (to USEL FIFO B)
:12313
:12314 .7. Set up the CSR to branch to the correct IDC microcode loop.
:12315
:12316 .*****
:12317 .* IDC *
:12318 .*****
:12319 .The IDC microcode will continue looping on the following
:12320 .steps until MAX L asserts or a CLEAR IDC is issued.
:12321
:12322 .IDC3. Select BEN1/MAX L.
:12323 .IDC4. If MAX L is asserted then assert XFER REQ and go to
:12324 .DIAG.WAIT.
:12325 .IDC5. Go to IDC3.
:12326
:12327 .8. Clear the FIFO ADRS CNTR.
:12328 .9. Check that XFER REQ is unasserted. (XFER REQ will assert if MAX L is
:12329 .asserted)
:12330
:12331 .10. Increment the error number.
:12332 .11. Select an R80 drive.
:12333 .12. Set WR[3] = 0 to use as a counter.
:12334 .13. Write a byte of data to FIFO B to increment the FIFO ADRS CNTR by 1.
:12335 .14. Increment WR[3].
:12336 .15. Check that XFER REQ is unasserted.
:12336 .16. Repeat Steps 13 - 16 until WR[3] = 510 (1FE HEX).
```

:12337 : 17. Increment the error number.  
:12338 : 18. Write a byte of data to FIFO B to increment the FIFO ADRS CNTR by 1.  
:12339 : 19. Increment WR[3].  
:12340 : 20. Check that XFER REQ is asserted.(XFER REQ asserts if MAX L asserted)  
:12341 : 21. CLEAR IDC.  
:12342 : 22. End test.

:12343 :  
:12344 :  
:12345 : Errors:

:12346 : Printout:

| EXPECTED | RECEIVED | OTHER DATA |
|----------|----------|------------|
| N/A      | N/A      | WR[3]      |

:12347 :  
:12348 :  
:12349 : ERROR 1 - PCLR FIFO CNTR failed or BEN1/FIFO MAX L failed unasserted.

:12350 :  
:12351 : ERROR 2 - PINCR FIFO CNTR failed or MAX L asserted too soon for an  
:12352 : R80 drive.

:12353 :  
:12354 : ERROR 3 - MAX L did not assert at 511 for an R80 drive.

:12355 :  
:12356 :  
:12357 : Debug:

:12358 : An R80 disk drive will be selected which will deassert L RL02 H at  
:12359 : the CSR SELECTED STATUS LATCH PAL.

:12360 : L RL02 H = L is inverted to become R80 H = H.

:12361 : ERROR 1:

:12362 : The CPU will issue PORT.CONTROL [CLEAR.FIFO.CNTR] which will  
:12363 : be decoded by the PORT INTERFACE REG PAL.  
:12364 : The inputs to the PORT INTERFACE REG PAL will be as follows:

:12365 :  
:12366 : CSR 17 H = H  
:12367 : CSR 14 H = H  
:12368 : CSR 12 H = L  
:12369 : CSR 11 H = L  
:12370 : CSR 10 H = L  
:12371 : PORT INSTR H = H  
:12372 : CPU P2 H = H

:12373 : When CPU CLOCK H clocks the PAL, PCLR FIFO CNTR H will assert. This  
:12374 : signal is an input to the FIFO ADRS CNTR PALS and will clear all of  
:12375 : the address lines and deassert FIFO MAX L.

:12376 : When the BEN1/FIFO.MAX.L IDC microinstruction is executed:

:12377 : BEN1 S0 H = H  
:12378 : BEN1 S1 H = L  
:12379 : BEN1 S2 H = H

:12380 : This will select FIFO MAX L as NUA 1 H. The IDC microcode will  
:12381 : remain in the loop waiting for FIFO MAX L to assert and will  
:12382 : not assert XFER REQ H.

:12383 : The CPU will then check to insure that XFER REQ H did not  
:12384 : assert by executing a SKIP.IF[NO.FAST.INTERRUPT].

:12385 :  
:12386 :  
:12387 : ERROR 2:





|   |   |
|---|---|
| U | 0 |
| U | 0 |
| U | 0 |

U O

U O

U O

50  
40

U O

U O

100

100

00  
40[illegible]

00

```

MOV LS[BEGIN.TEST] TO WR[0]
MOV WR[0] TO LS[CONTROL.STATUS]

MISC [SET.CP.ATTN]
2.0:
JMP [WAIT.T12.0]

;SET BIT 15 IN WR[0] FOR
; CONTROL AND STATUS WORD
;SET BIT 15 IN CONTROL AND
; STATUS WORD - BIT 15
; INDICATES START OF TEST TO
; CONSOLE
;ISSUE CPU ATTN TO CONSOLE

;LOOP HERE WAITING FOR
; CONSOLE TO RESPOND

```

C 6

ZZ-ENKCF-1.4 : ENKCF.MIC Test 12 FIFO B ADR15-MAR-83 Fiche 2 Frame C6 Sequence 273  
 : ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40 Page 267  
 : ENKCF.MIC Test 12 FIFO B ADRS CNTR MAX L BRANCH TEST 2 (M8388 IDC MODULE)

```

U 0C23, 65A0,15 :12502 CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
 :12503 ; IN LS
 :12504
U 0C24, B64A,15 :12505 MOV LS[IDC] TO WR[0] ;STORE MODULE CODE IN LS TO
U 0C25, 3E8C,15 :12506 MOV WR[0] TO LS[MODULE.NUM] ; INDICATE IDC MODULE
 :12507
 :12508
 :12509
U 0C26, F58A,15 :12510 CLR LS[ERROR.MASK]
 :12511
U 0C27, B670,15 :12512 MOV LS[OTHER.DATA] TO WR[0] ;SET UP TO PRINT OTHER DATA
U 0C28, 3E80,15 :12513 MOV WR[0] TO LS[CONTROL.STATUS] ; IF ERROR
 :12514
U 0C29, B66E,15 :12515 MOV LS[NA.EXP.REC] TO WR[0] ;SET UP TO PRINT N/A FOR
U 0C2A, 6D80,15 :12516 BIS WR[0] TO LS[CONTROL.STATUS] ; EXPECTED & RECEIVED DATA
 :12517
 :12518
 :12519
U 0C2B, B78E,15 :12520 MOV LS[DRIVE.STATUS] TO WR[0] ;HAS THE DRIVE SIZING ROUTINE
 :12521 BIT LS[BIT7] WITH WR[0], ; BEEN RUN?
U 0C2C, D94E,35 :12522 DT(LONG)&SET.ALU.CC
U 0C2D, 08C2,F1 :12523 JMP.IF[BIT.SET] TO [SIZE.OK.T12.1] ;IF YES THEN GO RUN TEST
 :12524
U 0C2E, 8870,0C :12525 JSR [DRIVE.SIZE] ;GO RUN DRIVE SIZING ROUTINE
 :12526
 :12527 SIZE.OK.T12.1:
U 0C2F, 8872,EC :12528
 :12529 JSR [DIAG.CODE] ;FORCE THE IDC TO DIAG.IDLE
 :12530
 :12531 ;SELECT AN R80 DRIVE IF ONE IS AVAILABLE...ELSE SKIP TEST
 :12532
U 0C30, B78E,15 :12533 MOV LS[DRIVE.STATUS] TO WR[0]
U 0C31, 3790,95 :12534 MOV LS[FFFFFFFF] TO WR[1]
U 0C32, AF42,15 :12535 BIC WR[1] TO WR[0] ;CHECK BITS <3:0>
 :12536 CMP LS[ZERO] WITH WR[0], ;ANY R80S?
 :12537 DT(LONG)&SET.ALU.CC
U 0C33, CE9C,35 :12538 JMP.IF[EQL] TO [CLEANUP.T12.1] ;NO R80, CLEANUP AND END TEST
U 0C34, 88C9,49 :12539
 :12540 MOV LS[DRIVE.STATUS] TO WR[0]
U 0C35, B78E,15 :12541 BIT LS[BIT0] WITH WR[0], ;IS DRIVE 0 AN R80?
 :12542 DT(LONG)&SET.ALU.CC
U 0C36, 5940,35 :12543 JMP.IF[BIT.SET] TO [SEL.DR0.T12.1] ;JMP IF DRV 0 = R80
U 0C37, 08C3,D1 :12544
 :12545 BIT LS[BIT1] WITH WR[0], ;IS DRIVE 1 AN R80?
 :12546 DT(LONG)&SET.ALU.CC
U 0C38, D942,35 :12547 JMP.IF[BIT.SET] TO [SEL.DR1.T12.1] ;JMP IF DRV 1 = R80
U 0C39, 88C3,F1 :12548
 :12549 BIT LS[BIT2] WITH WR[0], ;IS DRIVE 2 AN R80?
 :12550 DT(LONG)&SET.ALU.CC
U 0C3A, D944,35 :12551 JMP.IF[BIT.SET] TO [SEL.DR2.T12.1] ;JMP IF DRV 2 = R80
U 0C3B, 88C4,11 :12552 JMP [SEL.DR3.T12.1] ;JMP AND SEL DR 3
U 0C3C, 08C4,34 :12553
 :12554 SEL.DR0.T12.1:
U 0C3D, 3715,15 :12555 MOV LS[SETCSR.DS0] TO WR[2] ;SAV DRV SEL = 0
U 0C3E, 88C4,44 :12556 JMP [TEST.T12.1] ;GO DO TEST

```

```

:12557
:12558 SEL.DR1.T12.1:
U 0C3F, B717,15 :12559 MOV LS[SETCSR.DS1] TO WR[2] ;SAV DRV SEL = 1
U 0C40, 88C4,44 :12560 JMP [TEST.T12.1] ;GO DO TEST
:12561
:12562 SEL.DR2.T12.1:
U 0C41, 3719,15 :12563 MOV LS[SETCSR.DS2] TO WR[2] ;SAV DRV SEL = 2
U 0C42, 88C4,44 :12564 JMP [TEST.T12.1] ;GO DO TEST
:12565
:12566 SEL.DR3.T12.1:
U 0C43, B71B,15 :12567 MOV LS[SETCSR.DS3] TO WR[2] ;SAV DRV SEL = 3
:12568
:12569 TEST.T12.1:
U 0C44, 2F87,95 :12570 CLR WR[3] ;USE AS ADDR LOOP CNTR
:12571
:12572 PORT.CONTROL [SEL.FIFO.B] ;SEL FIFO B FROM THE PORT
U 0C45, 97DC,15 :12573 MOV LS[SETCSR.MAINT] TO WR[0] ;SET UP THE CSR TO BRANCH TO
U 0C46, 3720,15 :12574 MOV LS[SETCSR.F5] TO WR[1] ; THE CORRECT IDC ROUTINE
U 0C47, 370A,95 :12575 BIS WR[1] TO WR[0] ; TO SELECT FIFO A
U 0C48, AEC2,15 :12576 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
U 0C49, 17A0,15 :12577 ;MAINT=1, CRDY=0, F<2:0>=101
:12578
:12579 -----
:12580 *****
:12581 * IDC *
:12582 *****
:12583 -----
:12584 :018:
:12585 :=000
:12586 :DIAG.IDLE:
:12587
:12588 BEN0/F0,
:12589 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:12590 BEN2/F2,
:12591 NAD/DIAG.IDLE
:12592 -----
:12593 :01D:
:12594 :=101
:12595
:12596 BEN0/CRDY.L,
:12597 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:12598 NAD/TEST.FUNC5 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:12599 -----
:12600 :075:
:12601 :=1*1
:12602 :TEST.F5:
:12603
:12604 FUNC/CLR.ONLINE, ;CLR ONLINE FOR THE DRIVE SELECTED
:12605 ; BY DRIVE SEL<1:0> (ONLINE PAL TESTS)
:12606
:12607 UCMD/SET.FIFOA, ;SELECT FIFO A (USED FOR ANY TEST WHICH
:12608 ; MUST USEL FIFO A
:12609 NAD/XFER.REQ ;DO NOT INCR DAR BETWEEN HERE
:12610 ; AND DIAG.IDLE
:12611 -----

```

ZZ-1  
 :  
 :

U 0  
 U 0  
 U 0

U 0  
 U 0

U 0  
 U 0  
 U 0

```

12612 :1D8:
12613 :XFER.REQ:
12614 :
12615 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
12616 : NAD/DIAG.WAIT
12617 -----
12618 :1D7:
12619 :DIAG.WAIT:
12620 :
12621 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
12622 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
12623 -----
12624 :159:
12625 :=*0*
12626 :DIAG.WAIT1:
12627 :
12628 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
12629 : ; COMMAND FROM THE CPU
12630 -----
12631 :15B:
12632 :=*1*
12633 :DIAG.WAIT2:
12634 :
12635 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
12636 : ; WAIT SOME MORE
12637 -----
12638 :
12639 :
12640 :
12641 :WAIT.IDC.T12.1:
12642 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ TO SET
12643 : JMP [IDC.DONE.T12.1] ;XFER REQ ASSERTED
12644 : JMP [WAIT.IDC.T12.1] ;WAIT SOME MORE
12645 :
12646 :IDC.DONE.T12.1:
12647 : MOV LS[SETCSR.F0] TO WR[0] ;SET UP TO CLR CSR F<2:0>
12648 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
12649 : MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
12650 :
12651 : MOV LS[SETCSR.MAINT] TO WR[0] ;SET UP THE CSR TO BRANCH TO
12652 : MOV LS[SETCSR.F4] TO WR[1] ; THE IDC ROUTINE THAT WILL
12653 : BIS WR[1] TO WR[0] ; UCHANGE FIFO TO SEL FIFO B
12654 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
12655 : MAINT=1,CRDY=0, F<2:0>=100
12656 -----
12657 :
12658 :
12659 : *****
12660 : * IDC *
12661 : *****
12662 -----
12663 :018:
12664 :=*000
12665 :DIAG.IDLE:
12666 :

```

[illegible]

[illegible]

|   |   |
|---|---|
| U | 0 |
| U | 0 |

[illegible]

H 6

ZZ-ENKCF-1.4 : ENKCF.MIC Test 12 FIFO B ADR15-MAR-83 Fiche 2 Frame H6 Sequence 278  
 : ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40 Page 272  
 : ENKCF.MIC Test 12 FIFO B ADRS CNTR MAX L BRANCH TEST 2 (M8388 IDC MODULE)

```

U 0C64, 3792,15 :12777 MOV LS[#6] TO WR[0] ;NOP TO WAIT FOR IDC
 :12778 NOP.T12.1:
 :12779 DEC WR[0],
U 0C65, A100,35 :12780 DT(LONG)&SET.ALU.CC
U 0C66, 88C6,51 :12781 JMP.IF[NEQ] TO [NOP.T12.1]
 :12782
U 0C67, DB00,1A :12783 SKIP.IF[NO.FAST.INTERRUPT] ;SEE IF MAX L ASSERTED
U 0C68, 08C8,A4 :12784 JMP [ERROR.T12.1] ;MAX L ASSERTED WHEN FIFO
 :12785 ;ADRS WAS CLEAR
 :12786
U 0C69, 2F80,15 :12787 CLR WR[0]
U 0C6A, 2F82,95 :12788 CLR WR[1] ;CALL CHECK.RESULT TO LOCK
U 0C6B, BE89,95 :12789 MOV WR[3] TO LS[ADDRESS.DATA] ; IN INTERMITTENT ERRORS
U 0C6C, 0869,3C :12790 JSR [CHECK.RESULT]
U 0C6D, 88C5,A4 :12791 JMP [LOOP.T12.1]
 :12792
 :12793
U 0C6E, 3642,15 :12794 MOV LS[#2] TO WR[0] ;SET UP FOR ERROR 2
U 0C6F, BE82,15 :12795 MOV WR[0] TO LS[ERROR.NUMBER]
 :12796
 :12797
U 0C70, 378C,15 :12798 MOV LS[#1FE] TO WR[0] ;ONE BEFORE MAX CNT FOR AN R80
U 0C71, 3E10,15 :12799 MOV WR[0] TO LS[T8]
 :12800
U 0C72, 2F87,95 :12801 CLR WR[3] ;USE TO TRACK FIFO ADRS
 :12802 INCR.CNTR.T12.1:
U 0C73, 97A8,15 :12803 PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;WRITE A BYTE OF UNKNOWN DATA
 :12804 ; TO FIFO A TO INCR ADRS CNTR
U 0C74, 2047,95 :12805 INC WR[3] ; INCR ADRS TRACKER
U 0C75, B788,15 :12806 MOV LS[#3] TO WR[0] ;NOP TO WAIT FOR IDC
 :12807 NOP.T12.2:
 :12808 DEC WR[0],
U 0C76, A100,35 :12809 DT(LONG)&SET.ALU.CC
U 0C77, 08C7,61 :12810 JMP.IF[NEQ] TO [NOP.T12.2]
 :12811
 :12812
U 0C78, DB00,1A :12813 SKIP.IF[NO.FAST.INTERRUPT] ;CHECK IF MAX L ASSERTED
U 0C79, 08C8,A4 :12814 JMP [ERROR.T12.1] ;MAX L ASSERTED TOO SOON
 :12815
U 0C7A, 2F80,15 :12816 CLR WR[0] ;CALL CHECK.RESULT TO LOCK
U 0C7B, 2F82,95 :12817 CLR WR[1] ; IN INTERMITTENT ERRORS
U 0C7C, BE89,95 :12818 MOV WR[3] TO LS[ADDRESS.DATA]
U 0C7D, 0869,3C :12819 JSR [CHECK.RESULT]
U 0C7E, 88C5,A4 :12820 JMP [LOOP.T12.1]
 :12821
 :12822
 :12823
U 0C7F, 4E11,B5 :12824 CMP LS[T8] WITH WR[3], ;LAST COUNT?
U 0C80, 08C7,31 :12825 DT(LONG)&SET.ALU.CC
 :12826 JMP.IF[NEQ] TO [INCR.CNTR.T12.1] ;JUMP IF NOT LAST COUNT
U 0C81, B788,15 :12827 MOV LS[#3] TO WR[0]
U 0C82, BE82,15 :12828 MOV WR[0] TO LS[ERROR.NUMBER] ;SET UP FOR ERROR 3
 :12829
U 0C83, 97A8,15 :12830 PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;WRITE A BYTE OF DATA TO FIFO A
 :12831 ; TO INCR ADRS CNTR TO MAX (FF)

```

ZZ-E  
:  
:

U 0C

|              |   |           |         |        |                 |                                  |                                       |              |          |      |
|--------------|---|-----------|---------|--------|-----------------|----------------------------------|---------------------------------------|--------------|----------|------|
| ZZ-ENKCF-1.4 | : | ENKCF.MIC | Test 12 | FIFO B | ADR15-MAR-83    | Fiche 2                          | Frame 16                              | Sequence 279 | Page 273 | ZZ-E |
| :            | : | ENKCF.MCR | MICRO2  | 1M(01) | 15-MAR-83       | 11:46:22                         | ENKCF Micro-diagnostic for IDC module | Rev 01.40    |          | :    |
| :            | : | ENKCF.MIC | Test 12 | FIFO B | ADRS CNTR MAX L | BRANCH TEST 2 (M8388 IDC MODULE) |                                       |              |          | :    |

  

|                 |   |       |                               |  |                                 |
|-----------------|---|-------|-------------------------------|--|---------------------------------|
| U 0C84, B788,15 | : | 12832 | MOV LS[#3] TO WR[0]           |  | ;NOP TO WAIT FOR IDC            |
|                 | : | 12833 | NOP.T12.3:                    |  |                                 |
|                 | : | 12834 | DEC WR[0]                     |  |                                 |
| U 0C85, A100,35 | : | 12835 | DT(LONG)&SET.ALU.CC           |  |                                 |
| U 0C86, 08C8,51 | : | 12836 | JMP.IF[NEQ] TO [NOP.T12.3]    |  |                                 |
|                 | : | 12837 |                               |  |                                 |
| U 0C87, 2047,95 | : | 12838 | INC WR[3]                     |  | ;INCR THE ADRS TRACKER          |
| U 0C88, DB00,1A | : | 12839 | SKIP.IF[NO.FAST.INTERRUPT]    |  | ;MAX L ASSERTED?                |
| U 0C89, 08C8,F4 | : | 12840 | JMP [CALL.CH.RESULT.T12.1]    |  | ;MAX L ASSERTED OK              |
|                 | : | 12841 | ERROR.T12.1:                  |  | ;MAX L DID NOT ASSERT           |
| U 0C8A, 3640,95 | : | 12842 | MOV LS[#1] TO WR[1]           |  | ;FORCE AN ERROR                 |
| U 0C8B, 2F80,15 | : | 12843 | CLR WR[0]                     |  |                                 |
| U 0C8C, BE89,95 | : | 12844 | MOV WR[3] TO LS[ADDRESS.DATA] |  | ;PRINT OTHER DATA               |
| U 0C8D, 0869,3C | : | 12845 | JSR [CHECK.RESULT]            |  |                                 |
| U 0C8E, 88C5,A4 | : | 12846 | JMP [LOOP.T12.1]              |  | ;LOOP HERE IF ERR & LOE IS SET  |
|                 | : | 12847 |                               |  | ;ELSE SKIP OTHER ERRORS AND END |
|                 | : | 12848 |                               |  | ; TEST                          |
|                 | : | 12849 | CALL.CH.RESULT.T12.1:         |  |                                 |
| U 0C8F, 2F80,15 | : | 12850 | CLR WR[0]                     |  | ;CALL CHECK.RESULT TO LOCK      |
| U 0C90, 2F82,95 | : | 12851 | CLR WR[1]                     |  | ; IN INTERMITTENT ERRORS        |
| U 0C91, BE89,95 | : | 12852 | MOV WR[3] TO LS[ADDRESS.DATA] |  |                                 |
| U 0C92, 0869,3C | : | 12853 | JSR [CHECK.RESULT]            |  |                                 |
| U 0C93, 88C5,A4 | : | 12854 | JMP [LOOP.T12.1]              |  |                                 |
|                 | : | 12855 |                               |  |                                 |
|                 | : | 12856 | CLEANUP.T12.1:                |  |                                 |
| U 0C94, 8875,3C | : | 12857 | JSR [CLEANUP]                 |  | ;CLEANUP THE CONDITIONS SET BY  |
|                 | : | 12858 |                               |  | ; THIS TEST                     |
|                 | : | 12859 | END.T12:                      |  | ;END TEST                       |
|                 | : | 12860 |                               |  |                                 |
|                 | : | 12861 |                               |  |                                 |
|                 | : | 12862 |                               |  |                                 |



```
:12863 .PAGE 'Test 13 FIFO A ADRS CNTR OVFLW L BRANCH TEST 1 (M8388 IDC MODULE)''
:12864 .++
:12865
:12866 .Test Description:
:12867
:12868 .This test is designed to insure that OVFLW L asserts on the correct
:12869 .FIFO A address count (100 HEX) for an RL02 disk drive.
:12870 .This test will also verify that OVFLW L clears when the CPU issues a
:12871 .CLEAR.FIFO.CNTR instruction. BEN1/OVFLW L branch condition will also
:12872 .be verified.
:12873 .The test will then verify that USET FIFO OVFLW works correctly.
:12874
:12875 .Assumptions:
:12876
:12877 .It is assumed that XFER.REQ works correctly.
:12878
:12879 .Test Steps:
:12880
:12881 .1. Set up error mask, error number and module number in LS
:12882 .(for error printouts) and clear the previous error number in LS.
:12883 .2. If the DRIVE SIZING TEST has not been run previously then
:12884 .go run the DISK DRIVE SIZING ROUTINE.
:12885 .3. Force the IDC microcode to DIAG.IDLE.
:12886 .4. PSEL FIFO A.
:12887 .5. Set up the CSR to branch to the correct IDC microcode routine.
:12888
:12889 .*****
:12890 .* IDC *
:12891 .*****
:12892
:12893 .IDC1. Select FIFO A, UCM = 5.
:12894
:12895 .6. Set up the CSR to branch to the correct IDC microcode loop.
:12896
:12897 .*****
:12898 .* IDC *
:12899 .*****
:12900 .The IDC microcode will continue looping on the following
:12901 .steps until OVFLW L asserts or a CLEAR IDC is issued.
:12902
:12903 .IDC2. Select BEN1/OVFLW L.
:12904 .IDC3. If OVFLW L is asserted then assert XFER REQ and go to
:12905 .DIAG.WAIT.
:12906 .IDC4. Go to IDC2.
:12907
:12908 .7. Clear the FIFO ADRS CNTR.
:12909 .8. If XFER REQ is asserted (OVFLW asserted) then issue error.
:12910 .9. Increment the error number.
:12911 .10. Select an RL02 drive or a line with no drive attached.
:12912 .11. Set WR[3] = 0 to use as a counter.
:12913 .12. Write a byte of data to FIFO A to increment the FIFO ADRS CNTR by 1.
:12914 .13. Increment WR[3].
:12915 .14. If XFER REQ is asserted (OVFLW asserted) then issue error.
:12916 .15. Repeat Steps 12 - 15 until WR[3] = 255 (FF HEX).
:12917 .16. Increment the error number.
```

ZZ-ENKCF-1.4  
 : ENKCF.MCR  
 : ENKCF.MIC

K 6  
 Test 13 FIFO A ADR15-MAR-83  
 MICRO2 1M(01) 15-MAR-83 11:46:22  
 Test 13 FIFO A ADRS CNTR OVFLW L BRANCH TEST 1 (M8388 IDC MODULE)

Fiche 2 Frame K6  
 ENKCF Micro-diagnostic for IDC module  
 Sequence 281  
 Rev 01.40  
 Page 275

ZZ-E  
 :  
 :

  

:12918  
:12919  
:12920  
:12921  
:12922  
:12923  
:12924  
:12925  
:12926  
:12927  
:12928  
:12929  
:12930  
:12931  
:12932  
:12933  
:12934  
:12935  
:12936  
:12937  
:12938  
:12939  
:12940  
:12941  
:12942  
:12943  
:12944  
:12945  
:12946  
:12947  
:12948  
:12949  
:12950  
:12951  
:12952  
:12953  
:12954  
:12955  
:12956  
:12957  
:12958  
:12959  
:12960  
:12961  
:12962  
:12963  
:12964  
:12965  
:12966  
:12967  
:12968  
:12969  
:12970  
:12971  
:12972

17. Write a byte of data to FIFO A to increment the FIFO ADRS CNTR by 1.  
 18. Increment WR[3].  
 19. If XFER REQ is not asserted (OVFLW did not assert) then issue error.  
 20. Clear CSR F<2:0> and send XFER GRANT to clear the XFER REQ.  
 21. Increment the error number.  
 22. Force the IDC microcode to DIAG.IDLE.  
 23. Clear the FIFO ADRS CNTR.  
 24. Set up the CSR to branch to the correct IDC routine.  
  

\*\*\*\*\*  
 \* IDC \*  
 \*\*\*\*\*

  
 IDC5. Set FUNC<35:32> = 0E, USET.FIFO OVFLW.  
 IDC6. Perform Steps IDC2 - IDC4.  
  
 25. NOP to allow IDC time to branch.  
 26. If XFER REQ is not asserted (OVFLW did not assert) then issue error.  
 27. CLEAR IDC.  
 28. End test.  
  
 Errors:  
  

Printout:

| EXPECTED | RECEIVED | OTHER DATA |
|----------|----------|------------|
| N/A      | N/A      | WR[3]      |

  
 ERROR 1 - PCLR FIFO CNTR failed or BEN1/FIFO OVFLW L failed unasserted.  
 ERROR 2 - PINCR FIFO CNTR failed or OVFLW L asserted too soon for an RL02 drive.  
 ERROR 3 - OVFLW L did not assert at 256 for an RL02 drive.  
 ERROR 4 - USET FIFO OVFLW did not set OVFLW L.  
  
 Debug:  
  
 A disk drive that will assert L RL02 H at the CSR SELECTED  
 STAU5 LATCH PAL will be selected.  
 L RL02 H is inverted to become R80 H = L.  
  
 ERROR 1:  
  
 The CPU will issue PORT.CONTROL [CLEAR.FIFO.CNTR] which will  
 be decoded by the PORT INTERFACE REG PAL.  
 The inputs to the PORT INTERFACE REG PAL will be as follows:  
  

CSR 17 H = H  
 CSR 14 H = H  
 CSR 12 H = L

:12973 : CSR 11 H = L  
:12974 : CSR 10 H = L  
:12975 : PORT INSTR H = H  
:12976 : CPU P2 H = H  
:12977 : When CPU CLOCK H clocks the PAL, PCLR FIFO CNTR H will assert. This  
:12978 : signal is an input to the FIFO ADRS CNTR PALS and will clear  
:12979 : FIFO OVFLW L.  
:12980 :  
:12981 : When the BEN1/FIFO.OVFLW.L IDC microinstruction is executed:  
:12982 : BEN1 S0 H = L  
:12983 : BEN1 S1 H = L  
:12984 : BEN1 S2 H = H  
:12985 :  
:12986 : This will select FIFO OVFLW L as NUA 1 H. The IDC microcode will  
:12987 : remain in the loop waiting for FIFO OVFLW L to assert and will  
:12988 : not assert XFER REQ H.  
:12989 : The CPU will then check to insure that XFER REQ H did not  
:12990 : assert by executing a SKIP.IF[NO.FAST.INTERRUPT].  
:12991 :  
:12992 : ERROR 2:  
:12993 :  
:12994 : FIFO A is selected by the port by a PORT.CONTROL [SEL.FIFO.A]  
:12995 : instruction so that the FIFO CNTR can be incremented from the  
:12996 : PORT. This instruction is decoded by the PORT INTERFACE REG PAL.  
:12997 : The inputs will be:  
:12998 : CSR 17 H = H  
:12999 : CSR 14 H = H  
:13000 : CSR 12 H = H  
:13001 : CSR 11 H = H  
:13002 : CSR 10 H = L  
:13003 : PORT INSTR H = H  
:13004 : CPU P2 H = H  
:13005 : When the PAL is clocked by CPU CLOCK H, PSEL FIFO A H will assert.  
:13006 :  
:13007 : FIFO A will also be selected by the IDC microcode to allow FIFO A  
:13008 : OVFLW L to be asserted.  
:13009 : The IDC microcode will issue UCMD/SET.FIFOA. This will  
:13010 : assert UCMD 2 H and UCMD 0 H of the IDC microword. These bits are  
:13011 : decoded by the CONTROL STATUS REG PAL. The inputs will be:  
:13012 : UCMD 2 H = H  
:13013 : UCMD 1 H = L  
:13014 : UCMD 0 H = H  
:13015 : When the PAL is clocked by P2 CLOCK L, USEL FIFOB H will deassert and  
:13016 : USEL FIFOA H being the inverse of USEL FIFOB H, will assert.  
:13017 :  
:13018 : A PORT.WRITE PORT.ADRS[DATA.BYTE] is issued from the CPU. This  
:13019 : instruction will be decoded by the PORT RD/WR DECODE PAL.  
:13020 : The inputs to this PAL should be:  
:13021 : CSR 17 H = H  
:13022 : CSR 14 H = L  
:13023 : CSR 13 H = H  
:13024 : CSR 12 H = L  
:13025 : CSR 11 H = H  
:13026 : CSR 10 H = L  
:13027 : PORT INSTR H = H

```

:13028 : CPU P2 H = H
:13029 : This combination of inputs will assert BYTE L and WRITE DATA L, and
:13030 : STROBE DATA H.
:13031 : BYTE L and WRITE DATA L are inputs to the PORT USEQ A & B PALS and are
:13032 : clocked by CPU CLOCK H. This combination of inputs will assert
:13033 : PINCR FIFO CNTR H. This signal is ANDED with PSEL FIFO A H = H and
:13034 : PORT CLOCK L as the clock input to the FIFO A ADRS CNTR HI
:13035 : and LO PALS. When these PALS are clocked, FIFOA ADRS <8:0> H will
:13036 : increment.
:13037 : R80 H will be = L so FIFOA OVFLW L will not assert until
:13038 : FIFOA ADRS<7:4> = H and FIFOA LO FULL H = H and USEL FIFO A H = H
:13039 : and then the PAL must be clocked on more time by writing one more byte
:13040 : of data to FIFO A to assert PINCR FIFO CNTR H. FIFO OVFLW L will assert
:13041 : 20 ns after FIFOA OVFLW L.

```

When the BEN1/FIFO.OVFLW.L IDC microinstruction is executed:

BEN1 S0 H = L  
BEN1 S1 H = L  
BEN1 S2 H = H

U ODI

U OD

This will select FIFO OVFLW L as NUA 1 H. The IDC microcode will remain in the loop waiting for FIFO OVFLW L to assert and will not assert XFER REQ H. The CPU will then check to insure that XFER REQ H did not assert by executing a SKIP.IF[NO.FAST.INTERRUPT].

U OD

U OD

**ERROR 3:**

FIFOA ADRS <7:0> will be = 11111111. The FIFOA ADRS CNTR PALs will be clocked one more time by writing a byte of data to FIFO A. This will clock the FIFOA ADRS CNTR HI PAL.  
The inputs to the FIFOA ADRS CNTR HI PAL will be:

U OD

U OD

R80 H = L

```

R80 H = L
FIFOA LO FULL H = H
USEL FIFOA H = H
FIFOA ADRS <7:4> = H

```

U OD

This combination of inputs will assert FIFOA OVFLW L. Approximately 20ns later OVFLW L will assert.

U OD

U OD

**When the BEN1/FIFO.OVFLW.L IDC microinstruction is executed:**

BEN1 S0 H = L  
BEN1 S1 H = L  
BEN1 S2 H = H

U OD

U OD

This will select FIFO OVFLW L as NUA 1 H. The IDC microcode will branch to an instruction that will assert XFER REQ H. The CPU will then check to insure that XFER REQ H asserted by executing a SKIP.IF[NO.FAST.INTERRUPT].

U OD

U OD

U OD

U OD

**ERROR 4:**

The FIFO A ADRS CNTR PAL will be cleared which will insure that FIFO OVFLW L is not asserted. The IDC microcode will then branch to an instruction of FUNC/SET.FIFO.OVF. The output of the microsequencer

U OD

[illegible]

22-ENKCF-1.4 :  
: ENKCF.MCR  
: ENKCF.MIC

ENKCF.MIC

MICRO2 1M(01)

Test 13 FIFO A ADR15-MAR-83

15-MAR-83 11:46:22

Fiche 2 Frame B7

Sequence 285

Rev 01.40

Page 279

Test 13 FIFO A ADRS CNTR OVFLW L BRANCH TEST 1 (M8388 IDC MODULE)

22-1  
:  
:

```
U OCA2, D94E,35 :13138 DT(LONG)&SET.ALU.CC
U OCA3, 88CA,51 :13139 JMP.IF[BIT.SET] TO [SIZE.OK.T13.1] ;IF YES THEN GO RUN TEST
:13140
U OCA4, 8870,0C :13141 JSR [DRIVE.SIZE] ;GO RUN DRIVE SIZING ROUTINE
:13142
:13143 SIZE.OK.T13.1:
:13144
U OCA5, 8872,EC :13145 JSR [DIAG.CODE] ;FORCE THE IDC TO DIAG.IDLE
:13146
:13147 ;SELECT AN RLO2 DRIVE
:13148
U OCA6, B78E,15 :13149 MOV LS[DRIVE.STATUS] TO WR[0]
:13150 BIT LS[BIT0] WITH WR[0], ;IS DRIVE 0 AN R80?
U OCA7, 5940,35 :13151 DT(LONG)&SET.ALU.CC
U OCA8, 88CA,B9 :13152 JMP.IF[BITS.CLR] TO [SET.DS0.T13.1] ;JMP IF DRV 0 IS NOT AN R80
U OCA9, B717,15 :13153 MOV LS[SETCSR.DS1] TO WR[2] ;SAVE THE DATA TO SEL DRIVE 1
U OCAA, 88CA,C4 :13154 JMP [TEST.T13.1]
:13155 SET.DS0.T13.1:
U OCAB, 3715,15 :13156 MOV LS[SETCSR.DS0] TO WR[2] ;SAVE THE DATA TO SEL DRIVE 0
:13157
:13158 TEST.T13.1:
:13159
U OCAC, 2F87,95 :13160 CLR WR[3] ;USE AS ADDR LOOP CNTR
:13161
U OCAD, 17D8,15 :13162 PORT.CONTROL [SEL.FIFO.A] ;SEL FIFO A FROM THE PORT
U OCAE, 3720,15 :13163 MOV LS[SETCSR.MAINT] TO WR[0] ;SET UP THE CSR TO BRANCH TO
U OCAF, 370A,95 :13164 MOV LS[SETCSR.F5] TO WR[1] ; THE CORRECT IDC ROUTINE
U OCBO, AEC2,15 :13165 BIS WR[1] TO WR[0] ; TO USEL FIFO A
U OCB1, 17A0,15 :13166 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:13167 ;MAINT=1, CRDY=0, F<2:0>=101
:13168 ;USEL FIFO A
:13169 -----
:13170 -----
:13171 :
:13172 : *****
:13173 : * IDC *
:13174 : *****
:13175 : -----
:13176 :018:
:13177 :=000
:13178 :DIAG.IDLE:
:13179 :
:13180 : BEN0/F0,
:13181 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:13182 : BEN2/F2,
:13183 : NAD/DIAG.IDLE
:13184 : -----
:13185 :01D:
:13186 :=101
:13187 :
:13188 : BEN0/CRDY.L,
:13189 : BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:13190 : NAD/TEST.FUNCS ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:13191 : -----
:13192 :075:
:13193 :=1*1
```

```

:13193 :TEST.F5:
:13194 :
:13195 : FUNC/CLR.ONLINE, ;CLR ONLINE FOR THE DRIVE SELECTED
:13196 : ; BY DRIVE SEL<1:0> (ONLINE PAL TESTS)
:13197 :
:13198 : UCMD/SET.FIFOA, ;SELECT FIFO A (USED FOR ANY TEST WHICH
:13199 : ; MUST USEL FIFO A)
:13200 : NAD/XFER.REQ ;DO NOT INCR DAR BETWEEN HERE
:13201 : ; AND DIAG.IDLE
:13202 :-----
:13203 :1D8:
:13204 :XFER.REQ:
:13205 :
:13206 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:13207 : NAD/DIAG.WAIT
:13208 :-----
:13209 :1D7:
:13210 :DIAG.WAIT:
:13211 :
:13212 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:13213 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:13214 :
:13215 :-----
:13216 :159:
:13217 :=*0*
:13218 :DIAG.WAIT1:
:13219 :
:13220 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:13221 : ; COMMAND FROM THE CPU
:13222 :
:13223 :-----
:13224 :15B:
:13225 :=*1*
:13226 :DIAG.WAIT2:
:13227 :
:13228 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:13229 : ; WAIT SOME MORE
:13230 :
:13231 :-----
:13232 :
:13233 :WAIT.IDC.T13.1:
:13234 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ TO SET
:13235 : JMP [IDC.DONE.T13.1] ;XFER REQ ASSERTED
:13236 : JMP [WAIT.IDC.T13.1] ;WAIT SOME MORE
:13237 :
:13238 :IDC.DONE.T13.1:
:13239 : MOV LS[SETCSR.F0] TO WR[0] ;SET UP TO CLR CSR F<2:0>
:13240 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:13241 : MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
:13242 :
:13243 :LOOP.T13.1:
:13244 : MOV LS[#1] TO WR[0]
:13245 : MOV WR[0] TO LS[ERROR.NUMBER] ;ERROR 1
:13246 :
:13247 : PORT.CONTROL [CLEAR.FIFO.CNTR] ;CLEAR FIFO A ADRS CNTR

```

U OCB2, DB00,1A  
 U OCB3, 08CB,54  
 U OCB4, 88CB,24

U OCB5, B700,15  
 U OCB6, 17A0,15  
 U OCB7, 90FA,15

U OCB8, B640,15  
 U OCB9, BE82,15  
 U OCBA, 17C0,15

U 01  
 U 0  
 U 0  
  
 U 0  
 U 0  
 U 0  
  
 U 0  
 U 0  
 U 0  
  
 U 0  
 U 0  
 U 0

ZZ-ENKCF-1.4 ;  
: ENKCF.MCR  
: ENKCF.MIC

ENKCF.MIC

MICRO2 1M(01)

Test 13 FIFO A ADR15-MAR-83

15-MAR-83 11:46:22

Fiche 2 Frame D7

Sequence 287

Rev 01.40

Page 281

Test 13 FIFO A ADRS CNTR OVFLW L BRANCH TEST 1 (M8388 IDC MODULE)

ZZ-1  
:  
:

```
:13248
U OCBB, 3712,95 :13249 MOV LS[SETCSR.CRDY] TO WR[1] ;SET UP THE CSR TO BRANCH TO
U OCBC, B706,15 :13250 MOV LS[SETCSR.F3] TO WR[0] ; THE CORRECT IDC ROUTINE THAT
U OCBD, AEC2,15 :13251 BIS WR[1] TO WR[0] ; WILL SET XFER REQ WHEN OVFLW
:13252 ; ASSERTS
U OCBE, AEC4,15 :13253 BIS WR[2] TO WR[0] ;SELECT AN RL02
U OCBF, 17A0,15 :13254 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:13255 ;MAINT=0, CRDY=1, F<2:0>=011
:13256 -----
:13257 -----
:13258 ;*****
:13259 ; * IDC *
:13260 ;*****
:13261 -----
:13262 :018:
:13263 :=000
:13264 DIAG.IDLE:
:13265
:13266 BEN0/F0,
:13267 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:13268 BEN2/F2,
:13269 NAD/DIAG.IDLE
:13270 -----
:13271 :01B:
:13272 :=011
:13273 BEN0/CRDY.L,
:13274 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:13275 NAD/TEST.FUNC3 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:13276 -----
:13277 :068:
:13278 :=0*0
:13279 TEST.FUNC3:
:13280
:13281 TEST.FIFO.OVFLW.0:
:13282 BEN1/FIFO.OVFLW.L, ;IS FIFO OVFLW ASSERTED?
:13283 NAD/TEST.FIFO.OVFLW.1
:13284 -----
:13285 :190:
:13286 :=0*
:13287 TEST.FIFO.OVFLW.1:
:13288
:13289 NAD/XFER.REQ ;OVFLW ASSERTED
:13290 -----
:13291 :192:
:13292 :=1*
:13293
:13294 NAD/TEST.FIFO.OVFLW.0 ;OVFLW NOT ASSERTED...LOOP
:13295 -----
:13296
:13297
U OCC0, 3792,15 :13298 MOV LS[#6] TO WR[0] ;NOP TO WAIT FOR IDC
:13299 NOP.T13.1:
:13300 DEC WR[0],
:13301 DT(LONG)&SET.ALU.CC
U OCC1, A100,35 :13301 JMP.IF[NEQ] TO [NOP.T13.1]
U OCC2, 08CC,11 :13302
```

U 01

U 0  
U 0

U 0  
U 0

U 0  
U 0  
U 0  
U 0  
U 0

U 0  
U 0

U 0  
U 0  
U 0

U 0

U 0

U 0

U 0

U 0

U 0

U 0

U 0



|              |   |           |         |        |                                                    |          |                                       |              |          |              |
|--------------|---|-----------|---------|--------|----------------------------------------------------|----------|---------------------------------------|--------------|----------|--------------|
| ZZ-ENKCF-1.4 | : | ENKCF.MIC | Test 13 | FIFO A | ADR15-MAR-83                                       | Fiche 2  | Frame E7                              | Sequence 288 | Page 282 | ZZ-ENKCF-1.4 |
| :            | : | ENKCF.MCR | MICRO2  | 1M(01) | 15-MAR-83                                          | 11:46:22 | ENKCF Micro-diagnostic for IDC module | Rev 01.40    |          | :            |
| :            | : | ENKCF.MIC | Test 13 | FIFO A | ADRS CNTR OVFLW L BRANCH TEST 1 (M8388 IDC MODULE) |          |                                       |              |          | :            |

  

|                 |   |       |                                            |  |  |  |                                 |  |  |      |
|-----------------|---|-------|--------------------------------------------|--|--|--|---------------------------------|--|--|------|
| U OCC3, DB00,1A | : | 13303 |                                            |  |  |  |                                 |  |  | U 01 |
| U OCC4, 88D0,44 | : | 13304 | SKIP.IF[NO.FAST.INTERRUPT]                 |  |  |  | ;SEE IF OVFLW L ASSERTED        |  |  | U 01 |
|                 | : | 13305 | JMP [ERROR.T13.1]                          |  |  |  | ;OVFLW L ASSERTED WHEN FIFO     |  |  | U 01 |
|                 | : | 13306 |                                            |  |  |  | ;ADRS WAS CLEAR                 |  |  | U 01 |
| U OCC5, 2F80,15 | : | 13307 | CLR WR[0]                                  |  |  |  | ;CALL CHECK.RESULT TO LOCK      |  |  | U 01 |
| U OCC6, 2F82,95 | : | 13308 | CLR WR[1]                                  |  |  |  | ; IN INTERMITTENT ERRORS        |  |  | U 01 |
| U OCC7, BE89,95 | : | 13309 | MOV WR[3] TO LS[ADDRESS.DATA]              |  |  |  |                                 |  |  | U 01 |
| U OCC8, 0869,3C | : | 13310 | JSR [CHECK.RESULT]                         |  |  |  |                                 |  |  | U 01 |
| U OCC9, 88CB,84 | : | 13311 | JMP [LOOP.T13.1]                           |  |  |  |                                 |  |  |      |
|                 | : | 13312 |                                            |  |  |  |                                 |  |  |      |
| U OCCA, 3642,15 | : | 13313 | MOV LS[#2] TO WR[0]                        |  |  |  | ;SET UP FOR ERROR 2             |  |  | U 01 |
| U OCCB, BE82,15 | : | 13314 | MOV WR[0] TO LS[ERROR.NUMBER]              |  |  |  |                                 |  |  | U 01 |
|                 | : | 13315 |                                            |  |  |  |                                 |  |  |      |
|                 | : | 13316 |                                            |  |  |  |                                 |  |  | U 01 |
|                 | : | 13317 |                                            |  |  |  |                                 |  |  | U 01 |
| U OCCC, B626,15 | : | 13318 | MOV LS[0FF] TO WR[0]                       |  |  |  | ;ONE BEFORE OVFLW FOR AN RL02   |  |  |      |
| U OCCD, 3E10,15 | : | 13319 | MOV WR[0] TO LS[T8]                        |  |  |  |                                 |  |  | U 01 |
|                 | : | 13320 |                                            |  |  |  |                                 |  |  |      |
| U OCCE, 2F87,95 | : | 13321 | CLR WR[3]                                  |  |  |  | ;USE TO TRACK FIFO ADRS         |  |  | U 01 |
|                 | : | 13322 | INCR.CNTR.T13.1:                           |  |  |  |                                 |  |  | U 01 |
| U OCCF, 97A8,15 | : | 13323 | PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] |  |  |  | ;WRITE A BYTE OF UNKNOWN DATA   |  |  |      |
|                 | : | 13324 |                                            |  |  |  | ; TO FIFO A TO INCR ADRS CNTR   |  |  |      |
| U OCD0, 2047,95 | : | 13325 | INC WR[3]                                  |  |  |  | ;INCR ADRS TRACKER              |  |  | U 01 |
|                 | : | 13326 |                                            |  |  |  |                                 |  |  | U 01 |
| U OCD1, B788,15 | : | 13327 | MOV LS[#3] TO WR[0]                        |  |  |  | ;NOP TO WAIT FOR IDC            |  |  |      |
|                 | : | 13328 | NOP.T13.2:                                 |  |  |  |                                 |  |  |      |
|                 | : | 13329 | DEC WR[0]                                  |  |  |  |                                 |  |  | U 01 |
| U OCD2, A100,35 | : | 13330 | DT(LONG)&SET.ALU.CC                        |  |  |  |                                 |  |  | U 01 |
| U OCD3, 88CD,21 | : | 13331 | JMP.IF[NEQ] TO [NOP.T13.2]                 |  |  |  |                                 |  |  | U 0  |
|                 | : | 13332 |                                            |  |  |  |                                 |  |  |      |
|                 | : | 13333 |                                            |  |  |  |                                 |  |  |      |
| U OCD4, DB00,1A | : | 13334 | SKIP.IF[NO.FAST.INTERRUPT]                 |  |  |  | ;CHECK IF OVFLW L ASSERTED      |  |  | U 0  |
| U OCD5, 88D0,44 | : | 13335 | JMP [ERROR.T13.1]                          |  |  |  | ;OVFLW L ASSERTED TOO SOON      |  |  | U 0  |
|                 | : | 13336 |                                            |  |  |  |                                 |  |  | U 0  |
| U OCD6, 2F80,15 | : | 13337 | CLR WR[0]                                  |  |  |  | ;CALL CHECK.RESULT TO LOCK      |  |  | U 0  |
| U OCD7, 2F82,95 | : | 13338 | CLR WR[1]                                  |  |  |  | ; IN INTERMITTENT ERRORS        |  |  | U 0  |
| U OCD8, BE89,95 | : | 13339 | MOV WR[3] TO LS[ADDRESS.DATA]              |  |  |  |                                 |  |  | U 0  |
| U OCD9, 0869,3C | : | 13340 | JSR [CHECK.RESULT]                         |  |  |  |                                 |  |  |      |
| U OCDA, 88CB,84 | : | 13341 | JMP [LOOP.T13.1]                           |  |  |  |                                 |  |  | U 0  |
|                 | : | 13342 |                                            |  |  |  |                                 |  |  | U 0  |
|                 | : | 13343 |                                            |  |  |  |                                 |  |  |      |
|                 | : | 13344 | CMP LS[T8] WITH WR[3].                     |  |  |  | ;LAST COUNT?                    |  |  | U 0  |
| U OCDB, 4E11,B5 | : | 13345 | DT(LONG)&SET.ALU.CC                        |  |  |  |                                 |  |  | U 0  |
| U OCDC, 88CC,F1 | : | 13346 | JMP.IF[NEQ] TO [INCR.CNTR.T13.1]           |  |  |  | ;JUMP IF NOT LAST COUNT         |  |  | U 0  |
|                 | : | 13347 |                                            |  |  |  |                                 |  |  |      |
| U OCDD, B788,15 | : | 13348 | MOV LS[#3] TO WR[0]                        |  |  |  |                                 |  |  |      |
| U OCDE, BE82,15 | : | 13349 | MOV WR[0] TO LS[ERROR.NUMBER]              |  |  |  | ;SET UP FOR ERROR 3             |  |  | U 0  |
|                 | : | 13350 |                                            |  |  |  |                                 |  |  |      |
| U OCDF, 97A8,15 | : | 13351 | PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] |  |  |  | ;WRITE A BYTE OF DATA TO FIFO A |  |  |      |
|                 | : | 13352 |                                            |  |  |  | ; TO INCR ADRS CNTR TO OVFLW    |  |  |      |
| U OCE0, 2047,95 | : | 13353 | INC WR[3]                                  |  |  |  | ;INCR THE ADRS TRACKER          |  |  | U 0  |
| U OCE1, B788,15 | : | 13354 | MOV LS[#3] TO WR[0]                        |  |  |  | ;NOP TO WAIT FOR IDC            |  |  | U 0  |
|                 | : | 13355 | NOP.T13.3:                                 |  |  |  |                                 |  |  |      |
|                 | : | 13356 | DEC WR[0]                                  |  |  |  |                                 |  |  |      |
| U OCE2, A100,35 | : | 13357 | DT(LONG)&SET.ALU.CC                        |  |  |  |                                 |  |  |      |

|   |   |
|---|---|
| U | 0 |
| U | 0 |
| U | 0 |

U OC  
U OC

U 0C  
U 0C  
U 0C  
U 0C  
U 0C  
U 0C

U 01  
U 01  
U 01  
U 01  
U 01

U 02  
U 02  
U 02  
U 02  
U 02

U Of

U OD09, 8875,3C

:13468  
:13469  
:13470  
:13471  
:13472  
:13473  
:13474  
:13475

CLEANUP.T13.1:  
JSR [CLEANUP]  
  
END.T13:

;CLEANUP THE CONDITIONS SET BY  
; THIS TEST  
;END TEST

```
:13476 .PAGE 'Test 14 FIFO A ADRS CNTR OVFLW L BRANCH TEST 2 (M8388 IDC MODULE)''
:13477 .++
:13478
:13479 .Test Description:
:13480
:13481 . This test will only be run if an R80 is available on the system.
:13482 . This test is designed to verify that OVFLW L asserts on the correct
:13483 . address count (200 HEX) of FIFO A when an R80 is selected.
:13484 . This test will also verify USET OVFLW.
:13485
:13486 .Assumptions:
:13487
:13488 . It is assumed that XFER.REQ works correctly.
:13489
:13490 .Test Steps:
:13491
:13492 . 1. Set up error mask, error number and module number in LS
:13493 . (for error printouts) and clear the previous error number in LS.
:13494 . 2. If the DRIVE SIZING TEST has not been run previously then
:13495 . go run the DISK DRIVE SIZING ROUTINE.
:13496 . 3. If an R80 is not available on the system, then go to end of test.
:13497 . 4. Force the IDC microcode to DIAG.IDLE.
:13498 . 5. PSEL FIFO A.
:13499 . 6. Set up the CSR to branch to the correct IDC microcode routine.
:13500
:13501 . *****
:13502 . * IDC *
:13503 . *****
:13504
:13505 . IDC1. Select FIFO A, UCM = 5.
:13506
:13507 . 7. Set up the CSR to branch to the correct IDC microcode loop.
:13508
:13509 . *****
:13510 . * IDC *
:13511 . *****
:13512 . The IDC microcode will continue looping on the following
:13513 . steps until OVFLW L asserts or a CLEAR IDC is issued.
:13514
:13515 . IDC2. Select BEN1/OVFLW L.
:13516 . IDC3. If OVFLW L is asserted then assert XFER REQ and go to
:13517 . DIAG.WAIT.
:13518 . IDC4. Go to IDC2.
:13519
:13520 . 8. Clear the FIFO ADRS CNTR.
:13521 . 9. If XFER REQ is asserted (OVFLW asserted) then issue error.
:13522 . 10. Increment the error number.
:13523 . 11. Select an R80 drive.
:13524 . 12. Set WR[3] = 0 to use as a counter.
:13525 . 13. Write a byte of data to FIFO A to increment the FIFO ADRS CNTR by 1.
:13526 . 14. Increment WR[3].
:13527 . 15. Check that XFER REQ is unasserted.
:13528 . 16. Repeat Steps 13 - 16 until WR[3] = 511 (1FF HEX).
:13529 . 17. Increment the error number.
:13530 . 18. Write a byte of data to FIFO A to increment the FIFO ADRS CNTR by 1.
```

:13531 : 19. Increment WR[3].  
:13532 : 20. If XFER REQ is not asserted (OVFLW did not assert) then issue error.  
:13533 : 21. Force the IDC microcode to DIAG.IDLE.  
:13534 : 22. Clear the FIFO ADRS CNTR.  
:13535 : 23. Set up the CSR to branch to the correct IDC routine.

:13536 :  
:13537 : \*\*\*\*\*  
:13538 : \* IDC \*  
:13539 : \*\*\*\*\*

:13540 :  
:13541 : IDC5. Set FUNC<35:32> = 0E, USET.FIFO OVFLW.  
:13542 : IDC6. Perform Steps IDC2 - IDC4.

:13543 :  
:13544 : 24. NOP to allow IDC time to branch.  
:13545 : 25. If XFER REQ is not asserted (OVFLW did not assert) then issue error.  
:13546 : 26. CLEAR IDC.  
:13547 : 27. End test.

:13548 :  
:13549 : Errors:

:13550 :  
:13551 : Printout:

:13552 : EXPECTED  
:13553 : N/A

:13554 : RECEIVED  
:13555 : N/A

:13556 : OTHER DATA  
:13557 : WR[3]

:13558 : ERROR 1 - PCLR FIFO CNTR failed or BEN1/FIFO OVFLW L failed unasserted.

:13559 : ERROR 2 - PINCR FIFO CNTR failed or OVFLW L asserted too soon for an  
:13560 : R80 drive.

:13561 : ERROR 3 - OVFLW L did not assert at 512 for an R80 drive.

:13562 : ERROR 4 - OVFLW did not set when a USET FIFO OVFLW was executed.

:13563 :  
:13564 : Debug:

:13565 : An R80 disk drive will be selected which will deassert L RL02 H at  
:13566 : the CSR SELECTED STATUS LATCH PAL.  
:13567 : L RL02 H = L is inverted to become R80 H = H.

:13568 :  
:13569 : ERROR 1:

:13570 : The CPU will issue PORT.CONTROL [CLEAR.FIFO.CNTR] which will  
:13571 : be decoded by the PORT INTERFACE REG PAL.  
:13572 : The inputs to the PORT INTERFACE REG PAL will be as follows:

:13573 :  
:13574 : CSR 17 H = H  
:13575 : CSR 14 H = H  
:13576 : CSR 12 H = L  
:13577 : CSR 11 H = L  
:13578 : CSR 10 H = L  
:13579 : PORT INSTR H = H  
:13580 : CPU P2 H = H

:13581 : When CPU CLOCK H clocks the PAL, PCLR FIFO CNTR H will assert. This  
:13582 : signal is an input to the FIFO ADRS CNTR PALS and will clear  
:13583 : FIFO OVFLW L.  
:13584 :  
:13585 :

```

:13586 :
:13587 : When the BEN1/FIFO.OVFLW.L IDC microinstruction is executed:
:13588 : BEN1 S0 H = L
:13589 : BEN1 S1 H = L
:13590 : BEN1 S2 H = H
:13591 :
:13592 : This will select FIFO OVFLW L as NUA 1 H. The IDC microcode will
:13593 : remain in the loop waiting for FIFO OVFLW L to assert and will
:13594 : not assert XFER REQ H.
:13595 : The CPU will then check to insure that XFER REQ H did not
:13596 : assert by executing a SKIP.IF[NO.FAST.INTERRUPT].
:13597 :
:13598 : ERROR 2:
:13599 :
:13600 : FIFO A is selected by the port by a PORT.CONTROL [SEL.FIFO.A]
:13601 : instruction so that the FIFO CNTR can be incremented from the
:13602 : PORT. This instruction is decoded by the PORT INTERFACE REG PAL.
:13603 : The inputs will be:
:13604 : LSR 17 H = H
:13605 : CSR 14 H = H
:13606 : CSR 12 H = H
:13607 : CSR 11 H = H
:13608 : CSR 10 H = L
:13609 : PORT INSTR H = H
:13610 : CPU P2 H = H
:13611 : When the PAL is clocked by CPU CLOCK H, PSEL FIFO A H will assert.
:13612 :
:13613 : FIFO A will also be selected by the IDC microcode to allow FIFO A
:13614 : OVFLW L to be asserted.
:13615 : The IDC microcode will issue UCMD/SET.FIFOA. This will
:13616 : assert UCMD 2 H and UCMD 0 H of the IDC microword. These bits are
:13617 : decoded by the CONTROL STATUS REG PAL. The inputs will be:
:13618 : UCMD 2 H = H
:13619 : UCMD 1 H = L
:13620 : UCMD 0 H = H
:13621 : When the PAL is clocked by P2 CLOCK L, USEL FIFOB H will deassert and
:13622 : USEL FIFOA H being the inverse of USEL FIFOB H, will assert.
:13623 :
:13624 : A PORT.WRITE PORT.ADRS[DATA.BYTE] is issued from the CPU. This
:13625 : instruction will be decoded by the PORT RD/WR DECODE PAL.
:13626 : The inputs to this PAL should be:
:13627 : CSR 17 H = H
:13628 : CSR 14 H = L
:13629 : CSR 13 H = H
:13630 : CSR 12 H = L
:13631 : CSR 11 H = H
:13632 : CSR 10 H = L
:13633 : PORT INSTR H = H
:13634 : CPU P2 H = H
:13635 : This combination of inputs will assert BYTE L and WRITE DATA L, and
:13636 : STROBE DATA H.
:13637 : BYTE L and WRITE DATA L are inputs to the PORT USEQ A & B PALS and are
:13638 : clocked by CPU CLOCK H. This combination of inputs will assert
:13639 : PINCR FIFO CNTR H. This signal is ANDED with PSEL FIFO A H = H and
:13640 : PORT CLOCK L as the clock input to the FIFO A ADRS CNTR HI

```

:13641 : and LO PALs. When these PALs are clocked, FIFOA ADRL <8:0> H will  
:13642 : increment.  
:13643 :  
:13644 : R80 H will be = H so rIFOA OVFLW L will not assert until  
:13645 : FIFOA ADRL <8:4> = H and FIFOA LO FULL H = H and USEL FIFO A H = H and  
:13646 : then the PAL must be clocked one more time by writing a byte of data  
:13647 : to FIFO A to assert PINCR FIFO CNTR H. FIFO OVFLW L will assert  
:13648 : 20ns after FIFOA OVFLW L.

:13649 :  
:13650 : When the BEN1/FIFO.OVFLW.L IDC microinstruction is executed:

:13651 : BEN1 S0 H = L  
:13652 : BEN1 S1 H = L  
:13653 : BEN1 S2 H = H  
:13654 :

:13655 : This will select FIFO OVFLW L as NUA 1 H. The IDC microcode will  
:13656 : remain in the loop waiting for FIFO OVFLW L to assert and will  
:13657 : not assert XFER REQ H.  
:13658 : The CPU will then check to insure that XFER REQ H did not  
:13659 : assert by executing a SKIP.IF[NO.FAST.INTERRUPT].  
:13660 :

:13661 : ERROR 3:

:13662 :  
:13663 : FIFOA ADRL <8:0> will be = 11111111. The FIFOA ADRL CNTR PALs will  
:13664 : be clocked one more time by writing a byte of data to FIFO A.  
:13665 : The inputs to the FIFOA ADRL CNTR HI PAL will be:

:13666 : R80 H = H  
:13667 : FIFOA LO FULL H = H  
:13668 : USEL FIFOA H = H  
:13669 : FIFOA ADRL <8:4> = H  
:13670 :

:13671 : This combination of inputs will assert FIFOA OVFLW L. Approximately  
:13672 : 20ns later, FIFO OVFLW L will assert.

:13673 :  
:13674 : When the BEN1/FIFO.OVFLW.L IDC microinstruction is executed:

:13675 : BEN1 S0 H = L  
:13676 : BEN1 S1 H = L  
:13677 : BEN1 S2 H = H  
:13678 :

:13679 : This will select FIFO OVFLW L as NUA 1 H. The IDC microcode will  
:13680 : branch to an instruction that will assert XFER REQ H.  
:13681 : The CPU will then check to insure that XFER REQ H asserted by  
:13682 : executing a SKIP.IF[NO.FAST.INTERRUPT].  
:13683 :

:13684 : ERROR 4:

:13685 :  
:13686 : The FIFO A ADRL CNTR PAL will be cleared which will insure that  
:13687 : FIFO OVFLW L is not asserted. The IDC microcode will then branch to  
:13688 : an instruction of FUNC/SET.FIFO.OVF. The output of the microsequencer  
:13689 : will be:

:13690 : UFUNC 3 H = H  
:13691 : UFUNC 2 H = H  
:13692 : UFUNC 1 H = H  
:13693 : UFUNC 0 H = L  
:13694 :

:13695 : These bits will be decoded and USET FIFO OVF L will assert. This  
: signal is an input to the FIFOA ADRL CNTR HI PAL. USEL FIFOA H will

U OD

U OD

U OD

U OD

U OD

U OD

U OD



|                 |        |   |                                                                   |                               |      |
|-----------------|--------|---|-------------------------------------------------------------------|-------------------------------|------|
|                 | :13696 | : | also be asserted. When the PAL is clocked by INCR.FIFO.CNTR/ON,   |                               | U OD |
|                 | :13697 | : | FIFOA OVFLW L will assert. Approximately 20ns later, FIFO OVFLW L |                               | U OD |
|                 | :13698 | : | will assert.                                                      |                               | U OD |
|                 | :13699 | : |                                                                   |                               |      |
|                 | :13700 | : | When the BEN1/FIFO.OVFLW.L IDC microinstruction is executed:      |                               |      |
|                 | :13701 | : | BEN1 S0 H = L                                                     |                               | U OD |
|                 | :13702 | : | BEN1 S1 H = L                                                     |                               | U OD |
|                 | :13703 | : | BEN1 S2 H = H                                                     |                               |      |
|                 | :13704 | : |                                                                   |                               |      |
|                 | :13705 | : | This will select FIFO OVFLW L as NUA 1 H. The IDC microcode will  |                               | U OD |
|                 | :13706 | : | branch to an instruction that will assert XFER REQ H.             |                               |      |
|                 | :13707 | : | The CPU will then check to insure that XFER REQ H asserted by     |                               | U OD |
|                 | :13708 | : | executing a SKIP.IF[NO.FAST.INTERRUPT].                           |                               | U OD |
|                 | :13709 | : | --                                                                |                               |      |
|                 | :13710 | : | *****                                                             |                               | U OD |
|                 | :13711 | : | T.14:                                                             |                               |      |
| U ODOA, B65E,15 | :13712 | : | MOV LS[BEGIN.TEST] TO WR[0]                                       | ;SET BIT 15 IN WR[0] FOR      |      |
|                 | :13713 | : |                                                                   | ; CONTROL AND STATUS WORD     |      |
| U ODOB, 3E80,15 | :13714 | : | MOV WR[0] TO LS[CONTROL.STATUS]                                   | ;SET BIT 15 IN CONTROL AND    |      |
|                 | :13715 | : |                                                                   | ; STATUS WORD - BIT 15        | U OD |
|                 | :13716 | : |                                                                   | ; INDICATES START OF TEST TO  |      |
|                 | :13717 | : |                                                                   | ; CONSOLE                     |      |
| U ODOC, 10E0,15 | :13718 | : | MISC [SET.CP.ATTN]                                                | ;ISSUE CPU ATTN TO CONSOLE    |      |
|                 | :13719 | : | WAIT.T14.0:                                                       |                               | U OD |
| U ODOD, 88D0,D4 | :13720 | : | JMP [WAIT.T14.0]                                                  | ;LOOP HERE WAITING FOR        |      |
|                 | :13721 | : |                                                                   | ; CONSOLE TO RESPOND          | U OD |
|                 | :13722 | : |                                                                   |                               | U OD |
| U ODOE, 65A0,15 | :13723 | : | CLR LS[PREVIOUS.ERROR]                                            | ;CLEAR PREVIOUS ERROR NUMBER  | U OD |
|                 | :13724 | : |                                                                   | ; IN LS                       | U OD |
|                 | :13725 | : |                                                                   |                               |      |
| U OD0F, B64A,15 | :13726 | : | MOV LS[IDC] TO WR[0]                                              | ;STORE MODULE CODE IN LS TO   | U OD |
| U OD10, 3E8C,15 | :13727 | : | MOV WR[0] TO LS[MODULE.NUM]                                       | ; INDICATE IDC MODULE         |      |
|                 | :13728 | : |                                                                   |                               |      |
|                 | :13729 | : |                                                                   |                               |      |
|                 | :13730 | : |                                                                   |                               | U OD |
| U OD11, E58A,15 | :13731 | : | CLR LS[ERROR.MASK]                                                |                               |      |
|                 | :13732 | : |                                                                   |                               | U OD |
| U OD12, B670,15 | :13733 | : | MOV LS[OTHER.DATA] TO WR[0]                                       | ;SET UP TO PRINT OTHER DATA   |      |
| U OD13, 3E80,15 | :13734 | : | MOV WR[0] TO LS[CONTROL.STATUS]                                   | ; IF ERROR                    | U OD |
|                 | :13735 | : |                                                                   |                               | U OD |
| U OD14, B66E,15 | :13736 | : | MOV LS[NA.EXP.REC] TO WR[0]                                       | ;SET UP TO PRINT N/A FOR      | U OD |
| U OD15, 6D80,15 | :13737 | : | BIS WR[0] TO LS[CONTROL.STATUS]                                   | ; EXPECTED & RECEIVED DATA    | U OD |
|                 | :13738 | : |                                                                   |                               |      |
|                 | :13739 | : |                                                                   |                               |      |
| U OD16, B78E,15 | :13740 | : | MOV LS[DRIVE.STATUS] TO WR[0]                                     | ;HAS THE DRIVE SIZING ROUTINE |      |
|                 | :13741 | : | BIT LS[BIT7] WITH WR[0],                                          | ; RUN?                        |      |
| U OD17, D94E,35 | :13742 | : | DT(LONG)&SET.ALU.CC                                               |                               |      |
| U OD18, 88D1,A1 | :13743 | : | JMP.IF[BIT.SET] TO [SIZE.OK.T14.1]                                | ;IF YES THEN GO RUN TEST      |      |
|                 | :13744 | : |                                                                   |                               |      |
| U OD19, 8870,0C | :13745 | : | JSR [DRIVE.SIZE]                                                  | ;GO RUN DRIVE SIZING ROUTINE  |      |
|                 | :13746 | : |                                                                   |                               |      |
|                 | :13747 | : | SIZE.OK.T14.1:                                                    |                               |      |
|                 | :13748 | : |                                                                   |                               |      |
| U OD1A, 8872,EC | :13749 | : | JSR [DIAG.CODE]                                                   | ;FORCE THE IDC TO DIAG.IDLE   |      |
|                 | :13750 | : |                                                                   |                               |      |

|              |   |           |         |        |                   |               |                                       |              |          |       |
|--------------|---|-----------|---------|--------|-------------------|---------------|---------------------------------------|--------------|----------|-------|
| ZZ-ENKCF-1.4 | : | ENKCF.MIC | Test 14 | FIFO A | ADR15-MAR-83      | Fiche 2       | Frame N7                              | Sequence 297 | Page 291 | ZZ-El |
| :            | : | ENKCF.MIC | MICRO2  | 1M(01) | 15-MAR-83         | 11:46:22      | ENKCF Micro-diagnostic for IDC module | Rev 01.40    |          | :     |
| :            | : | ENKCF.MIC | Test 14 | FIFO A | ADRS CNTR OVFLW L | BRANCH TEST 2 | (M8388 IDC MODULE)                    |              |          | :     |

  

|                 |   |       |  |  |  |  |  |  |  |  |
|-----------------|---|-------|--|--|--|--|--|--|--|--|
| U OD1B, B78E,15 | : | 13751 |  |  |  |  |  |  |  |  |
| U OD1C, 3790,95 | : | 13752 |  |  |  |  |  |  |  |  |
| U OD1D, AF42,15 | : | 13753 |  |  |  |  |  |  |  |  |
| U OD1E, CE9C,35 | : | 13754 |  |  |  |  |  |  |  |  |
| U OD1F, 08D8,C9 | : | 13755 |  |  |  |  |  |  |  |  |
| U OD20, B78E,15 | : | 13756 |  |  |  |  |  |  |  |  |
| U OD21, 5940,35 | : | 13757 |  |  |  |  |  |  |  |  |
| U OD22, 08D2,81 | : | 13758 |  |  |  |  |  |  |  |  |
| U OD23, D942,35 | : | 13759 |  |  |  |  |  |  |  |  |
| U OD24, 88D2,A1 | : | 13760 |  |  |  |  |  |  |  |  |
| U OD25, D944,35 | : | 13761 |  |  |  |  |  |  |  |  |
| U OD26, 88D2,C1 | : | 13762 |  |  |  |  |  |  |  |  |
| U OD27, 08D2,E4 | : | 13763 |  |  |  |  |  |  |  |  |
| U OD28, 3715,15 | : | 13764 |  |  |  |  |  |  |  |  |
| U OD29, 88D2,F4 | : | 13765 |  |  |  |  |  |  |  |  |
| U OD2A, B717,15 | : | 13766 |  |  |  |  |  |  |  |  |
| U OD2B, 88D2,F4 | : | 13767 |  |  |  |  |  |  |  |  |
| U OD2C, 3719,15 | : | 13768 |  |  |  |  |  |  |  |  |
| U OD2D, 88D2,F4 | : | 13769 |  |  |  |  |  |  |  |  |
| U OD2E, B71B,15 | : | 13770 |  |  |  |  |  |  |  |  |
| U OD2F, 2F87,95 | : | 13771 |  |  |  |  |  |  |  |  |
| U OD30, 17D8,15 | : | 13772 |  |  |  |  |  |  |  |  |
| U OD31, 3720,15 | : | 13773 |  |  |  |  |  |  |  |  |
| U OD32, 370A,95 | : | 13774 |  |  |  |  |  |  |  |  |
| U OD33, AEC2,15 | : | 13775 |  |  |  |  |  |  |  |  |
| U OD34, 17A0,15 | : | 13776 |  |  |  |  |  |  |  |  |

  

|  |   |       |  |  |  |  |  |  |  |  |
|--|---|-------|--|--|--|--|--|--|--|--|
|  | : | 13751 |  |  |  |  |  |  |  |  |
|  | : | 13752 |  |  |  |  |  |  |  |  |
|  | : | 13753 |  |  |  |  |  |  |  |  |
|  | : | 13754 |  |  |  |  |  |  |  |  |
|  | : | 13755 |  |  |  |  |  |  |  |  |
|  | : | 13756 |  |  |  |  |  |  |  |  |
|  | : | 13757 |  |  |  |  |  |  |  |  |
|  | : | 13758 |  |  |  |  |  |  |  |  |
|  | : | 13759 |  |  |  |  |  |  |  |  |
|  | : | 13760 |  |  |  |  |  |  |  |  |
|  | : | 13761 |  |  |  |  |  |  |  |  |
|  | : | 13762 |  |  |  |  |  |  |  |  |
|  | : | 13763 |  |  |  |  |  |  |  |  |
|  | : | 13764 |  |  |  |  |  |  |  |  |
|  | : | 13765 |  |  |  |  |  |  |  |  |
|  | : | 13766 |  |  |  |  |  |  |  |  |
|  | : | 13767 |  |  |  |  |  |  |  |  |
|  | : | 13768 |  |  |  |  |  |  |  |  |
|  | : | 13769 |  |  |  |  |  |  |  |  |
|  | : | 13770 |  |  |  |  |  |  |  |  |
|  | : | 13771 |  |  |  |  |  |  |  |  |
|  | : | 13772 |  |  |  |  |  |  |  |  |
|  | : | 13773 |  |  |  |  |  |  |  |  |
|  | : | 13774 |  |  |  |  |  |  |  |  |
|  | : | 13775 |  |  |  |  |  |  |  |  |
|  | : | 13776 |  |  |  |  |  |  |  |  |
|  | : | 13777 |  |  |  |  |  |  |  |  |
|  | : | 13778 |  |  |  |  |  |  |  |  |
|  | : | 13779 |  |  |  |  |  |  |  |  |
|  | : | 13780 |  |  |  |  |  |  |  |  |
|  | : | 13781 |  |  |  |  |  |  |  |  |
|  | : | 13782 |  |  |  |  |  |  |  |  |
|  | : | 13783 |  |  |  |  |  |  |  |  |
|  | : | 13784 |  |  |  |  |  |  |  |  |
|  | : | 13785 |  |  |  |  |  |  |  |  |
|  | : | 13786 |  |  |  |  |  |  |  |  |
|  | : | 13787 |  |  |  |  |  |  |  |  |
|  | : | 13788 |  |  |  |  |  |  |  |  |
|  | : | 13789 |  |  |  |  |  |  |  |  |
|  | : | 13790 |  |  |  |  |  |  |  |  |
|  | : | 13791 |  |  |  |  |  |  |  |  |
|  | : | 13792 |  |  |  |  |  |  |  |  |
|  | : | 13793 |  |  |  |  |  |  |  |  |
|  | : | 13794 |  |  |  |  |  |  |  |  |
|  | : | 13795 |  |  |  |  |  |  |  |  |
|  | : | 13796 |  |  |  |  |  |  |  |  |
|  | : | 13797 |  |  |  |  |  |  |  |  |
|  | : | 13798 |  |  |  |  |  |  |  |  |
|  | : | 13799 |  |  |  |  |  |  |  |  |
|  | : | 13800 |  |  |  |  |  |  |  |  |
|  | : | 13801 |  |  |  |  |  |  |  |  |
|  | : | 13802 |  |  |  |  |  |  |  |  |
|  | : | 13803 |  |  |  |  |  |  |  |  |
|  | : | 13804 |  |  |  |  |  |  |  |  |
|  | : | 13805 |  |  |  |  |  |  |  |  |

  

|  |   |       |  |  |  |  |  |  |  |  |
|--|---|-------|--|--|--|--|--|--|--|--|
|  | : | 13751 |  |  |  |  |  |  |  |  |
|  | : | 13752 |  |  |  |  |  |  |  |  |
|  | : | 13753 |  |  |  |  |  |  |  |  |
|  | : | 13754 |  |  |  |  |  |  |  |  |
|  | : | 13755 |  |  |  |  |  |  |  |  |
|  | : | 13756 |  |  |  |  |  |  |  |  |
|  | : | 13757 |  |  |  |  |  |  |  |  |
|  | : | 13758 |  |  |  |  |  |  |  |  |
|  | : | 13759 |  |  |  |  |  |  |  |  |
|  | : | 13760 |  |  |  |  |  |  |  |  |
|  | : | 13761 |  |  |  |  |  |  |  |  |
|  | : | 13762 |  |  |  |  |  |  |  |  |
|  | : | 13763 |  |  |  |  |  |  |  |  |
|  | : | 13764 |  |  |  |  |  |  |  |  |
|  | : | 13765 |  |  |  |  |  |  |  |  |
|  | : | 13766 |  |  |  |  |  |  |  |  |
|  | : | 13767 |  |  |  |  |  |  |  |  |
|  | : | 13768 |  |  |  |  |  |  |  |  |
|  | : | 13769 |  |  |  |  |  |  |  |  |
|  | : | 13770 |  |  |  |  |  |  |  |  |
|  | : | 13771 |  |  |  |  |  |  |  |  |
|  | : | 13772 |  |  |  |  |  |  |  |  |
|  | : | 13773 |  |  |  |  |  |  |  |  |
|  | : | 13774 |  |  |  |  |  |  |  |  |
|  | : | 13775 |  |  |  |  |  |  |  |  |
|  | : | 13776 |  |  |  |  |  |  |  |  |
|  | : | 13777 |  |  |  |  |  |  |  |  |
|  | : | 13778 |  |  |  |  |  |  |  |  |
|  | : | 13779 |  |  |  |  |  |  |  |  |
|  | : | 13780 |  |  |  |  |  |  |  |  |
|  | : | 13781 |  |  |  |  |  |  |  |  |
|  | : | 13782 |  |  |  |  |  |  |  |  |
|  | : | 13783 |  |  |  |  |  |  |  |  |
|  | : | 13784 |  |  |  |  |  |  |  |  |
|  | : | 13785 |  |  |  |  |  |  |  |  |
|  | : | 13786 |  |  |  |  |  |  |  |  |
|  | : | 13787 |  |  |  |  |  |  |  |  |
|  | : | 13788 |  |  |  |  |  |  |  |  |
|  | : | 13789 |  |  |  |  |  |  |  |  |
|  | : | 13790 |  |  |  |  |  |  |  |  |
|  | : | 13791 |  |  |  |  |  |  |  |  |
|  | : | 13792 |  |  |  |  |  |  |  |  |
|  | : | 13793 |  |  |  |  |  |  |  |  |
|  | : | 13794 |  |  |  |  |  |  |  |  |
|  | : | 13795 |  |  |  |  |  |  |  |  |
|  | : | 13796 |  |  |  |  |  |  |  |  |
|  | : | 13797 |  |  |  |  |  |  |  |  |
|  | : | 13798 |  |  |  |  |  |  |  |  |
|  | : | 13799 |  |  |  |  |  |  |  |  |
|  | : | 13800 |  |  |  |  |  |  |  |  |
|  | : | 13801 |  |  |  |  |  |  |  |  |
|  | : | 13802 |  |  |  |  |  |  |  |  |
|  | : | 13803 |  |  |  |  |  |  |  |  |
|  | : | 13804 |  |  |  |  |  |  |  |  |
|  | : | 13805 |  |  |  |  |  |  |  |  |

  

|  |   |       |  |  |  |  |  |  |  |  |
|--|---|-------|--|--|--|--|--|--|--|--|
|  | : | 13751 |  |  |  |  |  |  |  |  |
|  | : | 13752 |  |  |  |  |  |  |  |  |
|  | : | 13753 |  |  |  |  |  |  |  |  |
|  | : | 13754 |  |  |  |  |  |  |  |  |
|  | : | 13755 |  |  |  |  |  |  |  |  |
|  | : | 13756 |  |  |  |  |  |  |  |  |
|  | : | 13757 |  |  |  |  |  |  |  |  |
|  | : | 13758 |  |  |  |  |  |  |  |  |
|  | : | 13759 |  |  |  |  |  |  |  |  |
|  | : | 13760 |  |  |  |  |  |  |  |  |
|  | : | 13761 |  |  |  |  |  |  |  |  |
|  | : | 13762 |  |  |  |  |  |  |  |  |
|  | : | 13763 |  |  |  |  |  |  |  |  |
|  | : | 13764 |  |  |  |  |  |  |  |  |
|  | : | 13765 |  |  |  |  |  |  |  |  |
|  | : | 13766 |  |  |  |  |  |  |  |  |
|  | : | 13767 |  |  |  |  |  |  |  |  |
|  | : | 13768 |  |  |  |  |  |  |  |  |
|  | : | 13769 |  |  |  |  |  |  |  |  |
|  | : | 13770 |  |  |  |  |  |  |  |  |
|  | : | 13771 |  |  |  |  |  |  |  |  |
|  | : | 13772 |  |  |  |  |  |  |  |  |
|  | : | 13773 |  |  |  |  |  |  |  |  |
|  | : | 13774 |  |  |  |  |  |  |  |  |
|  | : | 13775 |  |  |  |  |  |  |  |  |
|  | : | 13776 |  |  |  |  |  |  |  |  |
|  | : | 13777 |  |  |  |  |  |  |  |  |
|  | : | 13778 |  |  |  |  |  |  |  |  |
|  | : | 13779 |  |  |  |  |  |  |  |  |
|  | : | 13780 |  |  |  |  |  |  |  |  |
|  | : | 13781 |  |  |  |  |  |  |  |  |
|  | : | 13782 |  |  |  |  |  |  |  |  |
|  | : | 13783 |  |  |  |  |  |  |  |  |
|  | : | 13784 |  |  |  |  |  |  |  |  |
|  | : | 13785 |  |  |  |  |  |  |  |  |

```

:13806 :018:
:13807 :=000
:13808 :DIAG.IDLE:
:13809
:13810 BEN0/F0,
:13811 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:13812 BEN2/F2,
:13813 NAD/DIAG.IDLE
:13814 -----
:13815 :01D:
:13816 :=101
:13817
:13818 BEN0/CRDY.L,
:13819 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:13820 NAD/TEST.FUNC5 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:13821 -----
:13822 :075:
:13823 :=1*1
:13824 :TEST.F5:
:13825
:13826 FUNC/CLR.ONLINE, ;CLR ONLINE FOR THE DRIVE SELECTED
:13827 ; BY DRIVE SEL<1:0> (ONLINE PAL TESTS)
:13828
:13829 UCMD/SET.FIFOA, ;SELECT FIFO A (USED FOR ANY TEST WHICH
:13830 ; MUST USEL FIFO A
:13831 NAD/XFER.REQ ;DO NOT INCR DAR BETWEEN HERE
:13832 ; AND DIAG.IDLE
:13833 -----
:13834 :1D8:
:13835 :XFER.REQ:
:13836
:13837 UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:13838 NAD/DIAG.WAIT
:13839 -----
:13840 :1D7:
:13841 :DIAG.WAIT:
:13842
:13843 BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:13844 NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:13845 -----
:13846 :159:
:13847 :=*0*
:13848 :DIAG.WAIT1:
:13849
:13850 NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:13851 ; COMMAND FROM THE CPU
:13852 -----
:13853 :158:
:13854 :=*1*
:13855 :DIAG.WAIT2:
:13856
:13857 NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:13858 ; WAIT SOME MORE
:13859
:13860

```

U C  
U C  
U C  
U C  
U C  
U C

ZZ-ENKCF-1.4  
: ENKCF.MCR  
: ENKCF.MIC

ENKCF.MIC

MICRO2 1M(01)

Test 14 FIFO A ADR15-MAR-83

15-MAR-83 11:46:22

ENKCF Micro-diagnostic for IDC module

Sequence 299  
Rev 01.40

Page 293

Test 14 FIFO A ADRS CNTR OVFLW L BRANCH TEST 2 (M8388 IDC MODULE)

```

:13861 :
:13862 :-----
:13863 :
:13864 WAIT.IDC.T14.1:
U OD35, DB00,1A :13865 SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ TO SET
U OD36, 88D3,84 :13866 JMP [IDC.DONE.T14.1] ;XFER REQ ASSERTED
U OD37, 08D3,54 :13867 JMP [WAIT.IDC.T14.1] ;WAIT SOME MORE
:13868 :
:13869 IDC.DONE.T14.1:
U OD38, B700,15 :13870 MOV LS[SETCSR.F0] TO WR[0] ;SET UP TO CLR CSR F<2:0>
U OD39, 17A0,15 :13871 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
U OD3A, 90FA,15 :13872 MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
:13873 :
:13874 LOOP.T14.1:
:13875 :
U OD3B, B640,15 :13876 MOV LS[#1] TO WR[0] ;SET UP FOR ERROR 1
U OD3C, BE82,15 :13877 MOV WR[0] TO LS[ERROR.NUMBER]
:13878 :
U OD3D, 17C0,15 :13879 PORT.CONTROL [CLEAR.FIFO.CNTR] ;CLEAR FIFO A ADRS CNTR
:13880 :
U OD3E, 3712,95 :13881 MOV LS[SETCSR.CRDY] TO WR[1] ;SET UP THE CSR TO BRANCH TO
U OD3F, B706,15 :13882 MOV LS[SETCSR.F3] TO WR[0] ; THE CORRECT IDC ROUTINE
U OD40, AEC2,15 :13883 BIS WR[1] TO WR[0] ; THAT WILL SET XFER REQ WHEN
:13884 : ; OVFLW L ASSERTS
U OD41, AEC4,15 :13885 BIS WR[2] TO WR[0] ;SELECT AN R80
U OD42, 17A0,15 :13886 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:13887 : ;MAINT=0, CRDY=1, F<2:0>=011
:13888 :-----
:13889 :-----
:13890 : *****
:13891 : * IDC *
:13892 : *****
:13893 :-----
:13894 :018:
:13895 :=000
:13896 :DIAG.IDLE:
:13897 :
:13898 : BEN0/F0,
:13899 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:13900 : BEN2/F2,
:13901 : NAD/DIAG.IDLE
:13902 :-----
:13903 :018:
:13904 :=011
:13905 : BEN0/CRDY.L,
:13906 : BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:13907 : NAD/TEST.FUNC3 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:13908 :-----
:13909 :068:
:13910 :=0*0
:13911 :TEST.FUNC3:
:13912 :
:13913 :TEST.FIFO.OVLW.0:
:13914 : BEN1/FIFO.OVFLW.L, ;IS FIFO OVFLW ASSERTED?
:13915 : NAD/TEST.FIFO.OVFLW.1
```

```

:13916 :-----
:13917 :190:
:13918 :=0*
:13919 :TEST.FIFO.OVFLW.1:
:13920 :
:13921 : NAD/XFER.REQ :OVFLW ASSERTED
:13922 :-----
:13923 :192:
:13924 :=1*
:13925 :
:13926 : NAD/TEST.FIFO.OVFLW.0 :OVFLW NOT ASSERTED...LOOP
:13927 :-----
U OD43, 3792,15 :13928
:13929 MOV LS[#6] TO WR[0] ;NOP TO WAIT FOR IDC
:13930 NOP.T14.1:
:13931 DEC WR[0],
:13932 DT(LONG)&SET.ALU.CC
:13933 JMP.IF[NEQ] TO [NOP.T14.1]
:13934
:13935
:13936
U OD46, DB00,1A :13937
U OD47, 08D8,24 :13938 SKIP.IF[NO.FAST.INTERRUPT] ;SEE IF OVFLW L ASSERTED
:13939 JMP [ERROR.T14.1] ;OVFLW L ASSERTED WHEN FIFO
:13940 ; ADRS WAS CLEAR
:13941
U OD48, 2F80,15 :13941 CLR WR[0] ;CALL CHECK.RESULT TO LOCK
U OD49, 2F82,95 :13942 CLR WR[1] ; IN INTERMITTENT ERRORS
U OD4A, BE89,95 :13943 MOV WR[3] TO LS[ADDRESS.DATA]
U OD4B, 0869,3C :13944 JSR [CHECK.RESULT]
U OD4C, 88D3,B4 :13945 JMP [LOOP.T14.1]
:13946
:13947
U OD4D, 3642,15 :13948 MOV LS[#2] TO WR[0] ;SET UP FOR ERROR 2
U OD4E, BE82,15 :13949 MOV WR[0] TO LS[ERROR.NUMBER]
:13950
:13951
:13952
U OD4F, 378C,15 :13953 MOV LS[#1FE] TO WR[0]
U OD50, 2040,15 :13954 INC WR[0] ;ONE BEFORE OVFLW FOR AN R80
U OD51, 3E10,15 :13955 MOV WR[0] TO LS[T8]
:13956
:13957
U OD52, 2F87,95 :13957 CLR WR[3] ;USE TO TRACK FIFO ADRS
:13958 INCR.CNTR.T14.1:
U OD53, 97A8,15 :13959 PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;WRITE A BYTE OF UNKNOWN DATA
:13960 ; TO FIFO A TO INCR ADRS CNTR
U OD54, 2047,95 :13961 INC WR[3] ;INCR ADRS TRACKER
:13962
U OD55, B788,15 :13963 MOV LS[#3] TO WR[0] ;NOP TO WAIT FOR IDC
:13964 NOP.T14.2:
:13965 DEC WR[0],
:13966 DT(LONG)&SET.ALU.CC
:13967 JMP.IF[NEQ] TO [NOP.T14.2]
:13968
:13969
U OD58, DB00,1A :13970 SKIP.IF[NO.FAST.INTERRUPT] ;CHECK IF OVFLW L ASSERTED

```

E 8

ZZ-ENKCF-1.4 : ENKCF.MIC Test 14 FIFO A ADR15-MAR-83 Fiche 2 Frame E8 Sequence 301  
 : ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40 Page 295  
 : ENKCF.MIC Test 14 FIFO A ADRS CNTR OVFLW L BRANCH TEST 2 (M8388 IDC MODULE)

```

U OD59, 08D8,24 :13971 JMP [ERROR.T14.1] ;OVFLW L ASSERTED TOO SOON
 :13972
U OD5A, 2F80,15 :13973 CLR WR[0] ;CALL CHECK.RESULT TO LOCK
U OD5B, 2F82,95 :13974 CLR WR[1] ; IN INTERMITTENT ERRORS
U OD5C, BE89,95 :13975 MOV WR[3] TO LS[ADDRESS.DATA]
U OD5D, 0869,3C :13976 JSR [CHECK.RESULT]
U OD5E, 88D3,B4 :13977 JMP [LOOP.T14.1]
 :13978
 :13979
 :13980 CMP LS[T8] WITH WR[3], ;LAST COUNT?
U OD5F, 4E11,B5 :13981 DT(LONG)&SET.ALU.CC
U OD60, 08D5,31 :13982 JMP.IF[NEQ] TO [INCR.CNTR.T14.1] ;JUMP IF NOT LAST COUNT
 :13983
U OD61, B788,15 :13984 MOV LS[#3] TO WR[0]
U OD62, BE82,15 :13985 MOV WR[0] TO LS[ERROR.NUMBER] ;SET UP FOR ERROR 3
 :13986
U OD63, 97A8,15 :13987 PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;WRITE A BYTE OF DATA TO FIFO A
 :13988 ; TO INCR ADRS CNTR TO OVFLW
U OD64, 2047,95 :13989 INC WR[3] ;INCR THE ADRS TRACKER
 :13990
U OD65, B788,15 :13991 MOV LS[#3] TO WR[0] ;NOP TO WAIT FOR IDC
 :13992 NOP.T14.3:
 :13993 DEC WR[0],
U OD66, A100,35 :13994 DT(LONG)&SET.ALU.CC
U OD67, 08D6,61 :13995 JMP.IF[NEQ] TO [NOP.T14.3]
 :13996
 :13997
U OD68, DB00,1A :13998 SKIP.IF[NO.FAST.INTERRUPT] ;OVFLW L ASSERTED?
U OD69, 88D6,B4 :13999 JMP [OVFLW.OK.T14.1] ;OK
U OD6A, 08D8,24 :14000 JMP [ERROR.T14.1] ;OVFLW DID NOT ASSERT
 :14001
 :14002 OVFLW.OK.T14.1:
U OD6B, 2F80,15 :14003 CLR WR[0] ;CALL CHECK.RESULT TO LOCK
U OD6C, 2F82,95 :14004 CLR WR[1] ; IN INTERMITTENT ERRORS
U OD6D, BE89,95 :14005 MOV WR[3] TO LS[ADDRESS.DATA]
U OD6E, 0869,3C :14006 JSR [CHECK.RESULT]
U OD6F, 88D3,B4 :14007 JMP [LOOP.T14.1]
 :14008
U OD70, 3644,15 :14009 MOV LS[#4] TO WR[0]
U OD71, BE82,15 :14010 MOV WR[0] TO LS[ERROR.NUMBER] ;ERROR 4
 :14011
 :14012 LOOP.T14.2:
U OD72, 8872,EC :14013 JSR [DIAG.CODE] ;FORCE THE IDC TO DIAG.IDLE
U OD73, 17C0,15 :14014 PORT.CONTROL [CLEAR.FIFO.CNTR] ;CLEAR FIFO A ADRS CNTR
 :14015 ; TO DEASSERT OVFLW L
 :14016
 :14017
U OD74, B706,15 :14018 MOV LS[SETCSR.F3] TO WR[0] ;SET UP THE CSR TO BRANCH TO
 :14019 ; THE CORRECT IDC ROUTINE THAT
 :14020 ; WILL USE OVFLW AND THEN
 :14021 ; BRANCH TO SEE IF OVFLW DID
 :14022 ; ASSERT
U OD75, AEC4,15 :14022 BIS WR[2] TO WR[0] ;SELECT AN R80
U OD76, 17A0,15 :14023 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
 :14024 ;MAINT=0, CRDY=0, F<2:0>=011
 :14025 ;-----

```

```

:14026 :-----
:14027 : *****
:14028 : * IDC *
:14029 : *****
:14030 :-----
:14031 :018:
:14032 :=000
:14033 :DIAG.IDLE:
:14034 :
:14035 : BEN0/F0,
:14036 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:14037 : BEN2/F2,
:14038 : NAD/DIAG.IDLE
:14039 :-----
:14040 :01B:
:14041 :=011
:14042 : BEN0/CRDY.L,
:14043 : BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:14044 : NAD/TEST.FUNC3 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:14045 :-----
:14046 :069:
:14047 :=0*1
:14048 :TEST.USET.OVFLW:
:14049 :
:14050 : FUNC/SET.FIFO.OVF, ;USET OVFLW
:14051 : INCR.FIFO.CNTR/ON, ;CLK OVFLW
:14052 : NAD/TEST.FIFO.OVFLW.0
:14053 :-----
:14054 :068:
:14055 :=0*0
:14056 :TEST.FIFO.OVFLW.0:
:14057 :
:14058 : BEN1/FIFO.OVFLW.L, ;IS FIFO OVERFLOW ASSERTED?
:14059 : NAD/TEST.FIFO.OVFLW.1
:14060 :-----
:14061 :190:
:14062 :=0*
:14063 :TEST.FIFO.OVFLW.1:
:14064 :
:14065 : NAD/XFER.REQ ;OVFLW ASSERTED
:14066 :-----
:14067 :192:
:14068 :=1*
:14069 :
:14070 : NAD/TEST.FIFO.OVFLW.0 ;OVFLW NOT ASSERTED...LOOP
:14071 :-----
:14072 :
:14073 : MOV LS[#6] TO WR[0] ;NOP TO WAIT FOR IDC
:14074 :NOP.T14.4:
:14075 : DEC WR[0],
:14076 : DT(LONG)&SET.ALU.CC
:14077 : JMP.IF[NEQ] TO [NOP.T14.4]
:14078 :
:14079 :
:14080 :

```

U 0D77, 3792,15

U 0D78, A100,35

U 0D79, 08D7,81

ZZ-  
:  
:

U 0

U 0

U 0

|

[illegible]



```

:14117 .PAGE 'Test 15 FIFO B ADRS CNTR OVFLW L BRANCH TEST 1 (M8388 IDC MODULE)''
:14118 .++
:14119 .Test Description:
:14120 .
:14121 . This test is designed to insure that OVFLW L asserts on the correct
:14122 . FIFO B address count (100 HEX) for an RL02 disk drive.
:14123 . This test will also verify that OVFLW L clears when the CPU issues a
:14124 . CLEAR.FIFO.CNTR instruction.
:14125 . USET FIFO OVFLW will also be verified.
:14126 .
:14127 .Assumptions:
:14128 .
:14129 . It is assumed that XFER.REQ works correctly.
:14130 .
:14131 .Test Steps:
:14132 .
:14133 . 1. Set up error mask, error number and module number in LS
:14134 . (for error printouts) and clear the previous error number in LS.
:14135 . 2. If the DRIVE SIZING TEST has not been run previously then
:14136 . go run the DISK DRIVE SIZING ROUTINE.
:14137 . 3. Force the IDC microcode to DIAG.IDLE.
:14138 . 4. PSEL FIFO B.
:14139 . 5. Set up the CSR to branch to the correct IDC microcode routine.
:14140 .
:14141 . *****
:14142 . * IDC *
:14143 . *****
:14144 .
:14145 . IDC1. Select FIFO A, UCM = 5.
:14146 . IDC2. Change FIFO, UCM = 6. (to USEL FIFO B)
:14147 .
:14148 . 6. Set up the CSR to branch to the correct IDC microcode loop.
:14149 .
:14150 . *****
:14151 . * IDC *
:14152 . *****
:14153 . The IDC microcode will continue looping on the following
:14154 . steps until OVFLW L asserts or a CLEAR IDC is issued.
:14155 .
:14156 . IDC3. Select BEN1/OVFLW L.
:14157 . IDC4. If OVFLW L is asserted then assert XFER REQ and go to
:14158 . DIAG.WAIT.
:14159 . IDC5. Go to IDC3.
:14160 .
:14161 . 7. Clear the FIFO ADRS CNTR.
:14162 . 8. If XFER REQ is asserted (OVFLW asserted) then issue error.
:14163 . 9. Increment the error number.
:14164 . 10. Select an RL02 drive or a line with no drive attached.
:14165 . 11. Set WR[3], = 0 to use as a counter.
:14166 . 12. Write a byte of data to FIFO B to increment the FIFO ADRS CNTR by 1.
:14167 . 13. Increment WR[3].
:14168 . 14. If XFER REQ is asserted (OVFLW asserted) then issue error.
:14169 . 15. Repeat Steps 12 - 15 until WR[3] = 255 (FF HEX).
:14170 . 16. Increment the error number.
:14171 . 17. Write a byte of data to FIFO B to increment the FIFO ADRS CNTR by 1.

```

:14172 : 18. Increment WR[3].  
:14173 : 19. If XFER REQ is not asserted (OVFLW L did not assert) then  
:14174 : issue error.  
:14175 : 20. Clear CSR F<2:0> and send XFER GRANT to clear the XFER REQ.  
:14176 : 21. Increment the error number.  
:14177 : 22. Force the IDC microcode to DIAG.IDLE.  
:14178 : 23. Clear the FIFO ADRS CNTR.  
:14179 : 24. Set up the CSR to branch to the correct IDC routine.  
:14180 :  
:14181 : \*\*\*\*\*  
:14182 : \* IDC \*  
:14183 : \*\*\*\*\*  
:14184 :  
:14185 : IDC6. Set FUNC<35:32> = OE, USET.FIFO OVFLW.  
:14186 : IDC7. Perform Steps IDC3 - IDC5.  
:14187 :  
:14188 : 25. NOP to allow IDC time to branch.  
:14189 : 26. If XFER REQ is not asserted (OVFLW did not assert) then issue error.  
:14190 : 27. CLEAR IDC.  
:14191 : 28. End test.  
:14192 :  
:14193 :  
:14194 :  
:14195 :  
:14196 :  
:14197 :  
:14198 :  
:14199 :  
:14200 :  
:14201 :  
:14202 :  
:14203 :  
:14204 :  
:14205 :  
:14206 :  
:14207 :  
:14208 :  
:14209 :  
:14210 :  
:14211 :  
:14212 :  
:14213 :  
:14214 :  
:14215 :  
:14216 :  
:14217 :  
:14218 :  
:14219 :  
:14220 :  
:14221 :  
:14222 :  
:14223 :  
:14224 :  
:14225 :  
:14226 :

Errors:

Printout:

| EXPECTED                                                                         | RECEIVED | OTHER DATA |
|----------------------------------------------------------------------------------|----------|------------|
| N/A                                                                              | N/A      | WR[3]      |
| ERROR 1 - PCLR FIFO CNTR failed or BEN1/FIFO OVFLW L failed unasserted.          |          |            |
| ERROR 2 - PINCR FIFO CNTR failed or OVFLW L asserted too soon for an RL02 drive. |          |            |
| ERROR 3 - OVFLW L did not assert at 256 for an RL02 drive.                       |          |            |
| ERROR 4 - OVFLW did not set when a USET FIFO OVFLW was executed.                 |          |            |

Debug:

A disk drive that will assert L RL02 H at the CSR SELECTED  
STATUS LATCH PAL will be selected.  
L RL02 H is inverted to become R80 H = L.

ERROR 1:

The CPU will issue PORT.CONTROL [CLEAR.FIFO.CNTR] which will  
be decoded by the PORT INTERFACE REG PAL.  
The inputs to the PORT INTERFACE REG PAL will be as follows:

CSR 17 H = H  
CSR 14 H = H  
CSR 12 H = L  
CSR 11 H = L  
CSR 10 H = L

:14227 : PORT INSTR H = H  
:14228 : CPU P2 H = H  
:14229 : When CPU CLOCK H clocks the PAL, PCLR FIFO CNTR H will assert. This  
:14230 : signal is an input to the FIFO ADRS CNTR PALS and will clear  
:14231 : FIFO OVFLW L.  
:14232 :  
:14233 : When the BEN1/FIFO.OVFLW.L IDC microinstruction is executed:  
:14234 : BEN1 S0 H = L  
:14235 : BEN1 S1 H = L  
:14236 : BEN1 S2 H = H  
:14237 :  
:14238 : This will select FIFO OVFLW L as NUA 1 H. The IDC microcode will  
:14239 : remain in the loop waiting for FIFO OVFLW L to assert and will  
:14240 : not assert XFER REQ H.  
:14241 : The CPU will then check to insure that XFER REQ H did not  
:14242 : assert by executing a SKIP.IF[NO.FAST.INTERRUPT].  
:14243 :  
:14244 : ERROR 2:  
:14245 :  
:14246 : FIFO B is selected by the port by a PORT.CONTROL [SEL.FIFO.B]  
:14247 : instruction so that the FIFO CNTR can be incremented from the  
:14248 : PORT. This instruction is decoded by the PORT INTERFACE REG PAL.  
:14249 : The inputs will be:  
:14250 : CSR 17 H = H  
:14251 : CSR 14 H = H  
:14252 : CSR 12 H = H  
:14253 : CSR 11 H = H  
:14254 : CSR 10 H = H  
:14255 : PORT INSTR H = H  
:14256 : CPU P2 H = H  
:14257 :  
:14258 : When the PAL is clocked by CPU CLOCK H, PSEL FIFO A H will deassert.  
:14259 : PSEL FIFO B H = H, is the inverse of PSEL FIFO A H = L.  
:14260 :  
:14261 : FIFO B will also be selected by the IDC microcode to allow FIFO B  
:14262 : OVFLW L to be asserted.  
:14263 : The IDC microcode will issue UCMD/SET.FIFOA. This will  
:14264 : assert UCMD 2 H and UCMD 0 H of the IDC microword. These bits are  
:14265 : decoded by the CONTROL STATUS REG PAL. The inputs will be:  
:14266 : UCMD 2 H = H  
:14267 : UCMD 1 H = L  
:14268 : UCMD 0 H = H  
:14269 : When the PAL is clocked by P2 CLOCK L, USEL FIFOB H will deassert and  
:14270 : USEL FIFOA H being the inverse of USEL FIFOB H, will assert.  
:14271 : The IDC microcode will execute a UCMD/CHG.FIFO. This will assert  
:14272 : UCMD 2 H and UCMD 1 H of the IDC microword. These bits are decoded by  
:14273 : the CONTROL STATUS REG PAL. The inputs will be:  
:14274 : UCMD 2 H = H  
:14275 : UCMD 1 H = H  
:14276 : UCMD 0 H = L  
:14277 : When the PAL is clocked by P2 CLOCK L, USEL FIFOA H = L will be  
:14278 : inverted to USEL FIFOB H = H.  
:14279 :  
:14280 : A PORT.WRITE PORT.ADRS[DATA.BYTE] is issued from the CPU. This  
:14281 : instruction will be decoded by the PORT RD/WR DECODE PAL.

```

:14282 : The inputs to this PAL should be:
:14283 : CSR 17 H = H
:14284 : CSR 14 H = L
:14285 : CSR 13 H = H
:14286 : CSR 12 H = L
:14287 : CSR 11 H = H
:14288 : CSR 10 H = L
:14289 : PORT INSTR H = H
:14290 : CPU P2 H = H
:14291 : This combination of inputs will assert BYTE L and WRITE DATA L, and
:14292 : STROBE DATA H.
:14293 : BYTE L and WRITE DATA L are inputs to the PORT USEQ A & B PALS and are
:14294 : clocked by CPU CLOCK H. This combination of inputs will assert
:14295 : PINCR FIFO CNTR H. This signal is ANDED with PSEL FIFO B H = H and
:14296 : PORT CLOCK L as the clock input to the FIFO B ADRS CNTR HI
:14297 : and LO PALS. When these PALS are clocked, FIFOB ADRS<7:0> will
:14298 : increment.
:14299 :
:14300 : FIFO OVFLW L will not assert because the inputs to FIFOA ADRS CNTR HI
:14301 : PAL will be USEL FIFOA H = L and FIFOB OVFLW L = H.
:14302 : Since R80 H will be = L, FIFOB OVFLW L at the FIFOB ADRS CNTR HI PAL,
:14303 : will not assert until FIFOB ADRS<7:4> = H and FIFOB LO FULL H = H
:14304 : and USEL FIFO B H = H and the PAL is clocked one more time.
:14305 :
:14306 : When the BEN1/FIFO.OVFLW.L IDC microinstruction is executed:
:14307 : BEN1 S0 H = L
:14308 : BEN1 S1 H = L
:14309 : BEN1 S2 H = H
:14310 :
:14311 : This will select FIFO OVFLW L as NUA 1 H. The IDC microcode will
:14312 : remain in the loop waiting for FIFO OVFLW L to assert and will
:14313 : not assert XFER REQ H.
:14314 : The CPU will then check to insure that XFER REQ H did not
:14315 : assert by executing a SKIP.IF[NO.FAST.INTERRUPT].
:14316 :
:14317 : ERROR 3:
:14318 :
:14319 : FIFOB ADRS <7:0> will be = 11111111. The FIFOB ADRS CNTR PALS will
:14320 : be clocked one more time by writing a byte of data to FIFO B.
:14321 : The inputs to the FIFOB ADRS CNTR HI PAL will be:
:14322 : R80 H = L
:14323 : USEL FIFOB H = H
:14324 : FIFOB LO FULL H = H
:14325 : FIFOB ADRS <7:4> H = 1111
:14326 : This combination of inputs will assert FIFOB OVFLW L.
:14327 :
:14328 : The inputs to the FIFOA ADRS CNTR HI PAL will be:
:14329 : FIFOB OVFLW L = L
:14330 : USEL FIFO A H = L
:14331 : so approximately 20ns after FIFOB OVFLW L asserts, FIFO OVFLW L
:14332 : will assert.
:14333 :
:14334 : When the BEN1/FIFO.OVFLW.L IDC microinstruction is executed:
:14335 : BEN1 S0 H = L
:14336 : BEN1 S1 H = L

```

U 01  
U 01  
  
U 01  
U 01  
  
U 01  
  
U 01  
U 0  
  
U 01  
  
U 01  
U 0

[illegible]

```

14447 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
14448 : BEN2/F2,
14449 : NAD/DIAG.IDLE
14450 :-----
14451 : 01D:
14452 : =101
14453 :
14454 : BEN0/CRDY.L,
14455 : BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
14456 : NAD/TEST.FUNC5 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
14457 :-----
14458 : 075:
14459 : =1*1
14460 : TEST.F5:
14461 :
14462 : FUNC/CLR.ONLINE, ;CLR ONLINE FOR THE DRIVE SELECTED
14463 : ; BY DRIVE SEL<1:0> (ONLINE PAL TESTS)
14464 :
14465 : UCMD/SET.FIFOA, ;SELECT FIFO A (USED FOR ANY TEST WHICH
14466 : ; MUST USEL FIFO A
14467 : NAD/XFER.REQ ;DO NOT INCR DAR BETWEEN HERE
14468 : ; AND DIAG.IDLE
14469 :-----
14470 : 1D8:
14471 : XFER.REQ:
14472 :
14473 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
14474 : NAD/DIAG.WAIT
14475 :-----
14476 : 1D7:
14477 : DIAG.WAIT:
14478 :
14479 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
14480 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
14481 :
14482 :-----
14483 : 159:
14484 : =*0*
14485 : DIAG.WAIT1:
14486 :
14487 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
14488 : ; COMMAND FROM THE CPU
14489 :-----
14490 :
14491 : 158:
14492 : =*1*
14493 : DIAG.WAIT2:
14494 :
14495 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
14496 : ; WAIT SOME MORE
14497 :-----
14498 : WAIT.IDC.T15.1:
14499 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ TO SET
14500 : JMP [IDC.DONE.T15.1] ;XFER REQ ASSERTED
14501 : JMP [WAIT.IDC.T15.1] ;WAIT SOME MORE

```

U ODAA, DB00,1A  
 U ODAB, 88DA,D4  
 U ODAC, 08DA,A4

ZZ-E  
 :  
 :  
 U OE  
 U OE  
 U OE  
 U OE  
 U OE

ZZ-ENKCF-1.4 :  
: ENKCF.MCR  
: ENKCF.MIC

ENKCF.MIC

Test 15 FIFO B ADR15-MAR-83

Fiche 2 Frame B9

Sequence 311

Rev 01.40

Page 305

MICRO2 1M(01)

15-MAR-83 11:46:22

ENKCF Micro-diagnostic for IDC module

Test 15 FIFO B ADRS CNTR OVFLW L BRANCH TEST 1 (M8388 IDC MODULE)

```
:14502
:14503 IDC.DONE.T15.1:
U ODAD, B700,15 :14504 MOV LS[SETCSR.F0] TO WR[0] ;SET UP TO CLR CSR F<2:0>
U ODAE, 17A0,15 :14505 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
U ODAF, 90FA,15 :14506 MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
:14507
U ODB0, 3720,15 :14508 MOV LS[SETCSR.MAINT] TO WR[0] ;SET UP THE CSR TO BRANCH TO
U ODB1, B708,95 :14509 MOV LS[SETCSR.F4] TO WR[1] ; THE IDC ROUTINE THAT WILL
U ODB2, AEC2,15 :14510 BIS WR[1] TO WR[0] ; UCHANGE FIFO TO SEL FIFO B
U ODB3, 17A0,15 :14511 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:14512 ;MAINT=1,CRDY=0, F<2:0>=100
:14513 ;UCHNG FIFO TO SEL FIFO B
:14514 -----
:14515 -----
:14516 *****
:14517 * IDC *
:14518 *****
:14519 -----
:14520 :018:
:14521 =000
:14522 DIAG.IDLE:
:14523
:14524 BEN0/F0,
:14525 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:14526 BEN2/F2,
:14527 NAD/DIAG.IDLE
:14528 -----
:14529 :01C:
:14530 =100
:14531
:14532 BEN0/CRDY.L,
:14533 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:14534 NAD/TEST.FUNC4 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:14535 -----
:14536 :06F:
:14537 =1*1
:14538
:14539 TEST.F4:
:14540 UCMD/CHG.FIFO, ;CHANGE FIFO TO USEL FIFO B
:14541 INCR.DAR/ON, ;INCREMENT THE DAR
:14542 NAD/XFER.REQ ;SEND XFER REQ
:14543 -----
:14544 :1D8:
:14545 XFER.REQ:
:14546 UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:14547 NAD/DIAG.WAIT
:14548 -----
:14549 :1D7:
:14550 DIAG.WAIT:
:14551
:14552 BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:14553 NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:14554 -----
:14555 :159:
:14556 =*0*
```

ZZ  
:  
:

U  
U  
U  
U  
U  
U  
U  
U



nn nn nn nn nn

---

nn nn nn nn nn

```

14612 :-----
14613 :068:
14614 :=0*0
14615 :TEST.FUNC3:
14616 :
14617 :TEST.FIFO.OVFLW.0:
14618 : BEN1/FIFO.OVFLW.L, ;IS FIFO OVFLW ASSERTED?
14619 : NAD/TEST.FIFO.OVFLW.1
14620 :-----
14621 :190:
14622 :=0*
14623 :TEST.FIFO.OVFLW.1:
14624 :
14625 : NAD/XFER.REQ ;OVFLW ASSERTED
14626 :-----
14627 :192:
14628 :=1*
14629 :
14630 : NAD/TEST.FIFO.OVFLW.0 ;OVFLW NOT ASSERTED...LOOP
14631 :-----
U ODC2, 3792,15 14632
14633 : MOV LS[#6] TO WR[0] ;NOP TO WAIT FOR IDC
14634 : NOP.T15.1:
14635 : DEC WR[0],
14636 : DT(LONG)&SET.ALU.CC
14637 : JMP.IF[NEQ] TO [NOP.T15.1]
14638 :
14639 :
14640 :
14641 :
14642 : SKIP.IF[NO.FAST.INTERRUPT] ;SEE IF OVFLW L ASSERTED
14643 : JMP [ERROR.T15.1] ;OVFLW L ASSERTED WHEN FIFO
14644 : ;ADRS WAS CLEAR
14645 :
14646 : CLR WR[0] ;CALL CHECK.RESULT TO LOCK
14647 : CLR WR[1] ;IN INTERMITTENT ERRORS
14648 : MOV WR[3] TO LS[ADDRESS.DATA]
14649 : JSR [CHECK.RESULT]
14650 : JMP [LOOP.T15.1]
14651 :
14652 :
14653 :
14654 : MOV LS[#2] TO WR[0] ;SET UP FOR ERROR 2
14655 : MOV WR[0] TO LS[ERROR.NUMBER]
14656 :
14657 : MOV LS[#FF] TO WR[0] ;ONE BEFORE OVFLW FOR AN RL02
14658 : MOV WR[0] TO LS[T8]
14659 :
14660 : CLR WR[3] ;USE TO TRACK FIFO ADRS
14661 : INCR.CNTR.T15.1:
14662 : PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;WRITE A BYTE OF UNKNOWN DATA
14663 : ;TO FIFO B TO INCR ADRS CNTR
14664 : INC WR[3] ;INCR ADRS TRACKER
14665 :
14666 : MOV LS[#3] TO WR[0] ;NOP TO WAIT FOR IDC

```

```

:14667 NOP.T15.2:
:14668 DEC WR[0],
U ODD4, A100,35 :14669 DT(LONG)&SET.ALU.CC
U ODD5, 08DD,41 :14670 JMP.IF[NEQ] TO [NOP.T15.2]
:14671
:14672
U ODD6, DB00,1A :14673 SKIP.IF[NO.FAST.INTERRUPT] ;CHECK IF OVFLW L ASSERTED
U ODD7, 08E0,04 :14674 JMP [ERROR.T15.1] ;OVFLW L ASSERTED TOO SOON
:14675
:14676
U ODD8, 2F80,15 :14677 CLR WR[0] ;CALL CHECK.RESULT TO LOCK
U ODD9, 2F82,95 :14678 CLR WR[1] ; IN INTERMITTENT ERRORS
U ODDA, BE89,95 :14679 MOV WR[3] TO LS[ADDRESS.DATA]
U ODDB, 0869,3C :14680 JSR [CHECK.RESULT]
U ODDC, 88DB,A4 :14681 JMP [LOOP.T15.1]
:14682
:14683
U ODDD, 4E11,B5 :14684 CMP LS[T8] WITH WR[3], ;LAST COUNT?
U ODDE, 08DD,11 :14685 DT(LONG)&SET.ALU.CC ;JUMP IF NOT LAST COUNT
:14686
:14687
U ODDF, B788,15 :14688 MOV LS[#3] TO WR[0]
U ODE0, BE82,15 :14689 MOV WR[0] TO LS[ERROR.NUMBER] ;ERROR 3
:14690
U ODE1, 97A8,15 :14691 PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;WRITE A BYTE OF DATA TO
:14692 ;FIFO B
:14693 ; TO INCR ADRS CNTR TO OVFLW
U ODE2, 2047,95 :14694 INC WR[3] ;INCR THE ADRS TRACKER
U ODE3, B788,15 :14695 MOV LS[#3] TO WR[0] ;NOP TO WAIT FOR IDC
:14696
NOP.T15.3:
:14697 DEC WR[0],
U ODE4, A100,35 :14698 DT(LONG)&SET.ALU.CC
U ODE5, 08DE,41 :14699 JMP.IF[NEQ] TO [NOP.T15.3]
:14700
:14701
:14702
U ODE6, DB00,1A :14703 SKIP.IF[NO.FAST.INTERRUPT] ;OVFLW L ASSERTED?
U ODE7, 88DE,94 :14704 JMP [OVFLW.OK.T15.1] ;OK
U ODE8, 08E0,04 :14705 JMP [ERROR.T15.1] ;OVFLW DID NOT ASSERT
:14706
:14707
OVFLW.OK.T15.1:
:14708
U ODE9, 2F80,15 :14709 CLR WR[0] ;CALL CHECK.RESULT TO LOCK
U ODEA, 2F82,95 :14710 CLR WR[1] ; IN INTERMITTENT ERRORS
U ODEB, BE89,95 :14711 MOV WR[3] TO LS[ADDRESS.DATA]
U ODEC, 0869,3C :14712 JSR [CHECK.RESULT]
U ODED, 88DB,A4 :14713 JMP [LOOP.T15.1]
:14714
:14715
U ODEE, 3644,15 :14716 MOV LS[#4] TO WR[0]
U ODEF, BE82,15 :14717 MOV WR[0] TO LS[ERROR.NUMBER] ;ERROR 4
:14718
LOOP.T15.2:
U ODF0, 8872,EC :14718 JSR [DIAG.CODE] ;FORCE THE IDC TO DIAG.IDLE
U ODF1, 17C0,15 :14719 PORT.CONTROL [CLEAR.FIFO.CNTR] ;CLEAR FIFO B ADRS CNTR
:14720 ; TO DEASSERT OVFLW L
:14721

```

[illegible]

[illegible]

:14823 .PAGE "Test 16 FIFO B ADRS CNTR OVFLW L BRANCH TEST 2 (M8388 IDC MODULE)"

:14824 :++

:14825 :

:14826 :

:14827 :

:14828 :

:14829 :

:14830 :

:14831 :

:14832 :

:14833 :

:14834 :

:14835 :

:14836 :

:14837 :

:14838 :

:14839 :

:14840 :

:14841 :

:14842 :

:14843 :

:14844 :

:14845 :

:14846 :

:14847 :

:14848 :

:14849 :

:14850 :

:14851 :

:14852 :

:14853 :

:14854 :

:14855 :

:14856 :

:14857 :

:14858 :

:14859 :

:14860 :

:14861 :

:14862 :

:14863 :

:14864 :

:14865 :

:14866 :

:14867 :

:14868 :

:14869 :

:14870 :

:14871 :

:14872 :

:14873 :

:14874 :

:14875 :

:14876 :

:14877 :

# Test Description:

This test will only be run if an R80 is available on the system.  
 This test is designed to verify that OVFLW L asserts on the correct  
 address count (200 HEX) of FIFO B when an R80 is selected.  
 USET FIFO OVFLW will also be verified.

# Assumptions:

It is assumed that XFER.REQ works correctly.

# Test Steps:

1. Set up error mask, error number and module number in LS  
(for error printouts) and clear the previous error number in LS.
2. If the DRIVE SIZING TEST has not been run previously then  
go run the DISK DRIVE SIZING ROUTINE.
3. If an R80 is not available on the system, then go to end of test.
4. Force the IDC microcode to DIAG.IDLE.
5. PSEL FIFO B.
6. Set up the CSR to branch to the correct IDC microcode routine.

\*\*\*\*\*  
 \* IDC \*  
 \*\*\*\*\*

- IDC1. Select FIFO A, UCM = 5.
- IDC2. Change FIFO, UCM = 6. (to USEL FIFO B)

7. Set up the CSR to branch to the correct IDC microcode loop.

\*\*\*\*\*  
 \* IDC \*  
 \*\*\*\*\*

The IDC microcode will continue looping on the following  
steps until OVFLW L asserts or a CLEAR IDC is issued.

- IDC3. Select BEN1/OVFLW L.
- IDC4. If OVFLW L is asserted then assert XFER REQ and go to  
DIAG.WAIT.
- IDC5. Go to IDC3.

8. Clear the FIFO ADRS CNTR.
9. If XFER REQ is asserted (OVFLW asserted) then issue error.
10. Increment the error number.
11. Select an R80 drive.
12. Set WR[3] = 0 to use as a counter.
13. Write a byte of data to FIFO B to increment the FIFO ADRS CNTR by 1.
14. Increment WR[3].
15. If XFER REQ is asserted (OVFLW asserted) then issue error.
16. Repeat Steps 13 - 16 until WR[3] = 511 (1FF HEX).
17. Increment the error number.

:14878 : 18. Write a byte of data to FIFO B to increment the FIFO ADRS CNTR by 1.  
 :14879 : 19. Increment WR[3].  
 :14880 : 20. If XFER REQ is not asserted (OVFLW did not assert) then issue error.  
 :14881 : 21. Force the IDC microcode to DIAG.IDLE.  
 :14882 : 22. Clear the FIFO ADRS CNTR.  
 :14883 : 23. Set up the CSR to branch to the correct IDC routine.

\*\*\*\*\*  
 \* IDC \*  
 \*\*\*\*\*

IDC5. Set FUNC<35:32> = 0E, USET.FIFO OVFLW.  
 IDC6. Perform Steps IDC2 - IDC4.

:14892 : 24. NOP to allow IDC time to branch.  
 :14893 : 25. If XFER REQ is not asserted (OVFLW did not assert) then issue error.  
 :14894 : 26. CLEAR IDC.  
 :14895 : 27. End test.

:14898 : Errors:

| Printout:                                                                       | EXPECTED | RECEIVED | OTHER DATA |
|---------------------------------------------------------------------------------|----------|----------|------------|
|                                                                                 | N/A      | N/A      | WR[3]      |
| ERROR 1 - PCLR FIFO CNTR failed or BEN1/FIFO OVFLW L failed unasserted.         |          |          |            |
| ERROR 2 - PINCR FIFO CNTR failed or OVFLW L asserted too soon for an R80 drive. |          |          |            |
| ERROR 3 - OVFLW L did not assert at 512 for an R80 drive.                       |          |          |            |
| ERROR 4 - OVFLW did not set when a USET FIFO OVFLW was executed.                |          |          |            |

:14913 : Debug:  
 :14914 : An R80 disk drive will be selected which will deassert L RL02 H at  
 :14915 : the CSR SELECTED STATUS LATCH PAL.  
 :14916 : L RL02 H = L is inverted to become R80 H = H.

:14918 : ERROR 1:  
 :14919 :  
 :14920 : The CPU will issue PORT.CONTROL [CLEAR.FIFO.CNTR] which will  
 :14921 : be decoded by the PORT INTERFACE REG PAL.  
 :14922 : The inputs to the PORT INTERFACE REG PAL will be as follows:  
 :14923 :  
 :14924 : CSR 17 H = H  
 :14925 : CSR 14 H = H  
 :14926 : CSR 12 H = L  
 :14927 : CSR 11 H = L  
 :14928 : CSR 10 H = L  
 :14929 : PORT INSTR H = H  
 :14930 : CPU P2 H = H  
 :14931 : When CPU CLOCK H clocks the PAL, PCLR FIFO CNTR H will assert. This  
 :14932 : signal is an input to the FIFO ADRS CNTR PALS and will clear

:14933 :  
:14934 :  
:14935 :  
:14936 :  
:14937 :  
:14938 :  
:14939 :  
:14940 :  
:14941 :  
:14942 :  
:14943 :  
:14944 :  
:14945 :  
:14946 :  
:14947 :  
:14948 :  
:14949 :  
:14950 :  
:14951 :  
:14952 :  
:14953 :  
:14954 :  
:14955 :  
:14956 :  
:14957 :  
:14958 :  
:14959 :  
:14960 :  
:14961 :  
:14962 :  
:14963 :  
:14964 :  
:14965 :  
:14966 :  
:14967 :  
:14968 :  
:14969 :  
:14970 :  
:14971 :  
:14972 :  
:14973 :  
:14974 :  
:14975 :  
:14976 :  
:14977 :  
:14978 :  
:14979 :  
:14980 :  
:14981 :  
:14982 :  
:14983 :  
:14984 :  
:14985 :  
:14986 :  
:14987 :

FIFO OVFLW L.

When the BEN1/FIFO.OVFLW.L IDC microinstruction is executed:

BEN1 S0 H = L  
BEN1 S1 H = L  
BEN1 S2 H = H

This will select FIFO OVFLW L as NUA 1 H. The IDC microcode will remain in the loop waiting for FIFO OVFLW L to assert and will not assert XFER REQ H. The CPU will then check to insure that XFER REQ H did not assert by executing a SKIP.IF[NO.FAST.INTERRUPT].

ERROR 2:

FIFO B is selected by the port by a PORT.CONTROL [SEL.FIFO.B] instruction so that the FIFO CNTR can be incremented from the PORT. This instruction is decoded by the PORT INTERFACE REG PAL. The inputs will be:

CSR 17 H = H  
CSR 14 H = H  
CSR 12 H = H  
CSR 11 H = H  
CSR 10 H = H  
PORT INSTR H = H  
CPU P2 H = H

When the PAL is clocked by CPU CLOCK H, PSEL FIFO A H will deassert. PSEL FIFO B H = H, is the inverse of PSEL FIFO A H = L.

FIFO B will also be selected by the IDC microcode to allow FIFO B OVFLW L to be asserted.

The IDC microcode will issue UCMD/SET.FIFOA. This will assert UCMD 2 H and UCMD 0 H of the IDC microword. These bits are decoded by the CONTROL STATUS REG PAL. The inputs will be:

UCMD 2 H = H  
UCMD 1 H = L  
UCMD 0 H = H

When the PAL is clocked by P2 CLOCK L, USEL FIFOB H will deassert and USEL FIFOA H being the inverse of USEL FIFOB H, will assert.

The IDC microcode will execute a UCMD/CHG.FIFO. This will assert UCMD 2 H and UCMD 1 H of the IDC microword. These bits are decoded by the CONTROL STATUS REG PAL. The inputs will be:

UCMD 2 H = H  
UCMD 1 H = H  
UCMD 0 H = L

When the PAL is clocked by P2 CLOCK L, USEL FIFOA H = L will be inverted to USEL FIFOB H = H.

A PORT.WRITE PORT.ADRS[DATA.BYTE] is issued from the CPU. This instruction will be decoded by the PORT RD/WR DECODE PAL.

The inputs to this PAL should be:

CSR 17 H = H  
CSR 14 H = L  
CSR 13 H = H



MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module  
Test 16 FIFO B ADRS CNTR OVFLW L BRANCH TEST 2 (M8388 IDC MODULE)

```

14988 CSR 12 H = L
14989 CSR 11 H = H
14990 CSR 10 H = L
14991 PORT INSTR H = H
14992 CPU 02 H = H
14993 This combination of inputs will assert BYTE L and WRITE DATA L, and
14994 STROBE DATA H.
14995 BYTE L and WRITE DATA L are inputs to the PORT USEQ A & B PALs and are
14996 clocked by CPU CLOCK H. This combination of inputs will assert
14997 PINCR FIFO CNTR H. This signal is ANDED with PSEL FIFO B H = H and
14998 PORT CLOCK L as the clock input to the FIFO B ADRS CNTR HI
14999 and LO PALs. When these PALs are clocked, FIFOB ADRS<8:0> will
15000 increment.
15001
15002 FIFO OVFLW L will not assert because the inputs to FIFOA ADRS CNTR HI
15003 PAL will be USEL FIFOA H = L and FIFOB OVFLW L = H.
15004 Since R80 H will be = H, FIFOB OVFLW L at the FIFOB ADRS CNTR HI PAL,
15005 will not assert until FIFOB ADRS<8:4> = H and FIFOB LO FULL H = H
15006 and USEL FIFO B H = H and the PAL is clocked once more.
15007
15008 When the BEN1/FIFO.OVFLW.L IDC microinstruction is executed:
15009 BEN1 S0 H = L
15010 BEN1 S1 H = L
15011 BEN1 S2 H = H
15012
15013 This will select FIFO OVFLW L as NUA 1 H. The IDC microcode will
15014 remain in the loop waiting for FIFO OVFLW L to assert and will
15015 not assert XFER REQ H.
15016 The CPU will then check to insure that XFER REQ H did not
15017 assert by executing a SKIP.IF[NO.FAST.INTERRUPT].
15018
15019 ERROR 3:
15020 FIFOB ADRS <8:0> will be = 11111111. The FIFOB ADRS CNTR PALs will
15021 be clocked one more time by writing a byte of data to FIFO B.
15022 The inputs to the FIFOB ADRS CNTR HI PAL will be:
15023 R80 H = L
15024 USEL FIFOB H = H
15025 FIFOB LO FULL H = H
15026 FIFOB ADRS <7:4> H = 1111
15027 This combination of inputs will assert FIFOB OVFLW L.
15028
15029 The inputs to the FIFOA ADRS CNTR HI PAL will be:
15030 FIFOB OVFLW L = L
15031 USEL FIFO A H = L
15032 so approximately 20ns after FIFOB OVFLW L asserts, FIFO OVFLW L
15033 will assert.
15034
15035 When the BEN1/FIFO.OVFLW.L IDC microinstruction is executed:
15036 BEN1 S0 H = L
15037 BEN1 S1 H = L
15038 BEN1 S2 H = H
15039
15040 This will select FIFO OVFLW L as NUA 1 H. The IDC microcode will
15041 branch to an instruction that will assert XFER REQ H.
15042 The CPU will then check to insure that XFER REQ H asserted by

```

1 u c

100

100 (

! u c

U (

U (

U (

U (

```

:15043 : executing a SKIP.IF[NO.FAST.INTERRUPT].
:15044 :
:15045 : ERROR 4:
:15046 :
:15047 :
:15048 : The FIFO A ADRS CNTR PAL will be cleared which will insure that
:15049 : FIFO OVFLW L is not asserted. The IDC microcode will then branch to
:15050 : an instruction of FUNC/SET.FIFO.OVF. The output of the microsequencer
:15051 : will be:
:15052 : UFUNC 3 H = H
:15053 : UFUNC 2 H = H
:15054 : UFUNC 1 H = H
:15055 : UFUNC 0 H = L
:15056 : These bits will be decoded and USET FIFO OVF L will assert. This
:15057 : signal is an input to the FIFOB ADRS CNTR HI PAL. USEL FIFOB H will
:15058 : also be asserted. When the PAL is clocked by INCR.FIFO.CNTR/ON,
:15059 : FIFOB OVFLW L will assert. FIFOB OVFLW L = L is an input to the
:15060 : FIFOA ADRS CNTR H PAL. USEL FIFOA H will not be asserted, so
:15061 : approximately 20ns later, FIFO OVFLW L will assert.
:15062 :
:15063 : When the BEN1/FIFO.OVFLW.L IDC microinstruction is executed:
:15064 : BEN1 S0 H = L
:15065 : BEN1 S1 H = L
:15066 : BEN1 S2 H = H
:15067 :
:15068 : This will select FIFO OVFLW L as NUA 1 H. The IDC microcode will
:15069 : branch to an instruction that will assert XFER REQ H.
:15070 : The CPU will then check to insure that XFER REQ H asserted by
:15071 : --
:15072 : *****
:15073 : T.16:
:15074 : MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT 15 IN WR[0] FOR
:15075 : ; CONTROL AND STATUS WORD
:15076 : MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND
:15077 : ; STATUS WORD - BIT 15
:15078 : ; INDICATES START OF TEST TO
:15079 : ; CONSOLE
:15080 : MISC [SET.CP.ATTN] ;ISSUE CPU ATTN TO CONSOLE
:15081 : WAIT.T16.0:
:15082 : JMP [WAIT.T16.0] ;LOOP HERE WAITING FOR
:15083 : ; CONSOLE TO RESPOND
:15084 :
:15085 : CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
:15086 : ; IN LS
:15087 :
:15088 : MOV LS[IDC] TO WR[0] ;STORE MODULE CODE IN LS TO
:15089 : MOV WR[0] TO LS[MODULE.NUM] ; INDICATE IDC MODULE
:15090 :
:15091 :
:15092 :
:15093 : CLR LS[ERROR.MASK]
:15094 :
:15095 : MOV LS[OTHER.DATA] TO WR[0] ;SET UP TO PRINT OTHER DATA
:15096 : MOV WR[0] TO LS[CONTROL.STATUS] ; IF ERROR
:15097 :

```

M 9

ZZ-ENKCF-1.4 : ENKCF.MIC Test 16 FIFO B ADR15-MAR-83 Fiche 2 Frame M9 Sequence 322  
 : ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40 Page 316  
 : ENKCF.MIC Test 16 FIFO B ADRS CNTR OVFLW L BRANCH TEST 2 (M8388 IDC MODULE)

```

U 0E15, B66E,15 :15098 MOV LS[NA.EXP.REC] TO WR[0] ;SET UP TO PRINT N/A FOR
U 0E16, 6D80,15 :15099 BIS WR[0] TO LS[CONTROL.STATUS] ; EXPECTED & RECEIVED DATA
 :15100
 :15101
U 0E17, B78E,15 :15102 MOV LS[DRIVE.STATUS] TO WR[0] ;HAS THE DRIVE SIZING ROUTINE
 :15103 BIT LS[BIT7] WITH WR[0], ; BEEN RUN?
U 0E18, D94E,35 :15104 DT(LONG)&SET.ALU.CC
U 0E19, 08E1,B1 :15105 JMP.IF[BIT.SET] TO [SIZE.OK.T16.1] ;IF YES THEN GO RUN TEST
 :15106
U 0E1A, 8870,0C :15107 JSR [DRIVE.SIZE] ;GO RUN DRIVE SIZING ROUTINE
 :15108
 :15109 SIZE.OK.T16.1:
U 0E1B, 8872,EC :15111 JSR [DIAG.CODE] ;FORCE THE IDC TO DIAG.IDLE
 :15112
 :15113 ;SELECT AN R80 DRIVE IF ONE IS AVAILABLE...ELSE SKIP TEST
 :15114
U 0E1C, B78E,15 :15115 MOV LS[DRIVE.STATUS] TO WR[0]
U 0E1D, 3790,95 :15116 MOV LS[FFFFFFF0] TO WR[1]
U 0E1E, AF42,15 :15117 BIC WR[1] TO WR[0] ;CHECK BITS <3:0>
 :15118 CMP LS[ZERO] WITH WR[0], ;ANY R80S?
U 0E1F, CE9C,35 :15119 DT(LONG)&SET.ALU.CC
U 0E20, 08E9,79 :15120 JMP.IF[EQL] TO [CLEANUP.T16.1] ;NO R80, CLEANUP AND END TEST
 :15121
U 0E21, B78E,15 :15122 MOV LS[DRIVE.STATUS] TO WR[0]
 :15123 BIT LS[BIT0] WITH WR[0], ;IS DRIVE 0 AN R80?
U 0E22, 5940,35 :15124 DT(LONG)&SET.ALU.CC
U 0E23, 88E2,91 :15125 JMP.IF[BIT.SET] TO [SEL.DR0.T16.1] ;JMP IF DRV 0 = R80
 :15126
 :15127 BIT LS[BIT1] WITH WR[0], ;IS DRIVE 1 AN R80?
U 0E24, D942,35 :15128 DT(LONG)&SET.ALU.CC
U 0E25, 08E2,B1 :15129 JMP.IF[BIT.SET] TO [SEL.DR1.T16.1] ;JMP IF DRV 1 = R80
 :15130
 :15131 BIT LS[BIT2] WITH WR[0], ;IS DRIVE 2 AN R80?
U 0E26, D944,35 :15132 DT(LONG)&SET.ALU.CC
U 0E27, 08E2,D1 :15133 JMP.IF[BIT.SET] TO [SEL.DR2.T16.1] ;JMP IF DRV 2 = R80
 :15134
U 0E28, 88E2,F4 :15135 JMP [SEL.DR3.T16.1] ;JMP AND SEL DR 3
 :15136
 :15137 SEL.DR0.T16.1:
U 0E29, 3715,15 :15138 MOV LS[SETCSR.DS0] TO WR[2] ;SAV DRV SEL = 0
U 0E2A, 08E3,04 :15139 JMP [TEST.T16.1] ;GO DO TEST
 :15140
 :15141 SEL.DR1.T16.1:
U 0E2B, B717,15 :15142 MOV LS[SETCSR.DS1] TO WR[2] ;SAV DRV SEL = 1
U 0E2C, 08E3,04 :15143 JMP [TEST.T16.1] ;GO DO TEST
 :15144
 :15145 SEL.DR2.T16.1:
U 0E2D, 3719,15 :15146 MOV LS[SETCSR.DS2] TO WR[2] ;SAV DRV SEL = 2
U 0E2E, 08E3,04 :15147 JMP [TEST.T16.1] ;GO DO TEST
 :15148
 :15149 SEL.DR3.T16.1:
U 0E2F, B71B,15 :15150 MOV LS[SETCSR.DS3] TO WR[2] ;SAV DRV SEL = 3
 :15151
 :15152 TEST.T16.1:

```

ZZ-  
 :  
 :

U 0  
 U 0  
 U 0  
  
 U 0  
 U 0  
 U 0  
  
 U 0  
 U 0  
 U 0  
  
 U 0  
 U 0  
 U 0

```

U OE30, 2F87,95 :15153 CLR WR[3] ;USE AS ADDR TRACKER
:15154
U OE31, 97DC,15 :15155 PORT.CONTROL [SEL.FIFO.B] ;SEL FIFO B FROM THE PORT
U OE32, 3720,15 :15156 MOV LS[SETCSR.MAINT] ,O WR[0] ;SET UP THE CSR TO BRANCH TO
U OE33, 370A,95 :15157 MOV LS[SETCSR.F5] TO WR[1] ; THE CORRECT IDC ROUTINE
U OE34, AEC2,15 :15158 BIS WR[1] TO WR[0] ; TO SELECT FIFO A
U OE35, 17A0,15 :15159 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:15160 ;MAINT=1, CRDY=0, F<2:0>=101
:15161 ;USEL FIFO A
:15162 -----
:15163 -----
:15164 *****
:15165 * IDC *
:15166 *****
:15167 -----
:15168 018:
:15169 =000
:15170 DIAG.IDLE:
:15171
:15172 BEN0/F0,
:15173 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:15174 BEN2/F2,
:15175 NAD/DIAG.IDLE
:15176 -----
:15177 01D:
:15178 =101
:15179
:15180 BEN0/CRDY.L,
:15181 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:15182 NAD/TEST.FUNC5 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:15183 -----
:15184 075:
:15185 =1*1
:15186 TEST.F5:
:15187
:15188 FUNC/CLR.ONLINE, ;CLR ONLINE FOR THE DRIVE SELECTED
:15189 ; BY DRIVE SEL<1:0> (ONLINE PAL TESTS)
:15190
:15191 UCMD/SET.FIFOA, ;SELECT FIFO A (USED FOR ANY TEST WHICH
:15192 ; MUST USEL FIFO A
:15193 NAD/XFER.REQ ;DO NOT INCR DAR BETWEEN HERE
:15194 ; AND DIAG.IDLE
:15195 -----
:15196 1D8:
:15197 XFER.REQ:
:15198
:15199 UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:15200 NAD/DIAG.WAIT
:15201 -----
:15202 1D7:
:15203 DIAG.WAIT:
:15204
:15205 BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:15206 NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:15207

```

[illegible]

```

:15263 :06F:
:15264 :=1*1
:15265
:15266 :TEST.F4:
:15267 :UCMD/CHG.FIFO, ;CHANGE FIFO TO USEL FIFO B
:15268 :INCR.DAR/ON, ;INCREMENT THE DAR
:15269 :NAD/XFER.REQ ;SEND XFER REQ
:-----
:15270
:15271 :1D8:
:15272 :XFER.REQ:
:15273 :UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:15274 :NAD/DIAG.WAIT
:-----
:15275
:15276 :1D7:
:15277 :DIAG.WAIT:
:15278
:15279 :BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:15280 :NAD/DIAG.WAIT1 ;BEFORE RETURNING TO DIAG.IDLE
:-----
:15281
:15282 :159:
:15283 :=*0*
:15284 :DIAG.WAIT1:
:15285
:15286 :NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:15287 : ;COMMAND FROM THE CPU
:-----
:15288
:15289 :15B:
:15290 :=*1*
:15291 :DIAG.WAIT2:
:15292
:15293 :NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:15294 : ;WAIT SOME MORE
:-----
:15295
:15296 :WAIT.IDC.T16.2:
:15297 :SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ TO SET
:15298 :JMP [IDC.DONE.T16.2] ;XFER REQ ASSERTED
:15299 :JMP [WAIT.IDC.T16.2] ;WAIT SOME MORE
:15300 :IDC.DONE.T16.2:
:15301 :MOV LS[SETCSR.F0] TO WR[0] ;SET UP TO CLR CSR F<2:0>
:15302 :PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:15303 :MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
:15304
:15305
:15306 :LOOP.T16.1:
:15307 :MOV LS[#1] TO WR[0]
:15308 :MOV WR[0] TO LS[ERROR.NUMBER] ;ERROR 1
:15309
:15310 :PORT.CONTROL [CLEAR.FIFO.CNTR] ;CLEAR FIFO B ADRS CNTR
:15311
:15312 :MOV LS[SETCSR.CRDY] TO WR[1] ;SET UP THE CSR TO BRANCH TO
:15313 :MOV LS[SETCSR.F3] TO WR[0] ;THE IDC ROUTINE THAT WILL
:15314 :BIS WR[1] TO WR[0] ;LOOP UNTIL OVFLW ASSERTS
:15315 : ;AND THEN SETS XFER REQ
:15316 :BIS WR[2] TO WR[0] ;SELECT AN R80
:15317 :PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR

```

U OE40, DB00,1A  
 U OE41, 88E4,34  
 U OE42, 88E4,04  
 U OE43, B700,15  
 U OE44, 17A0,15  
 U OE45, 90FA,15

U OE46, B640,15  
 U OE47, BE82,15  
 U OE48, 17C0,15  
 U OE49, 3712,95  
 U OE4A, B706,15  
 U OE4B, AEC2,15

U OE4C, AFC4,15  
 U OE4D, 17A0,15

```

15318 ;MAINT=0, CRDY=1, F<2:0>=011
15319 -----
15320 -----
15321 *****
15322 * IDC *
15323 *****
15324 -----
15325 018:
15326 =000
15327 DIAG.IDLE:
15328
15329 BEN0/F0,
15330 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
15331 BEN2/F2,
15332 NAD/DIAG.IDLE
15333 -----
15334 01B:
15335 =011
15336 BEN0/CRDY.L,
15337 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
15338 NAD/TEST.FUNC3 ;DEPENDENT ON CSR <CRDY> AND <MAINT>
15339 -----
15340 068:
15341 =0*0
15342 TEST.FUNC3:
15343
15344 TEST.FIFO.OVFLW.0:
15345 BEN1/FIFO.OVFLW.L, ;IS FIFO OVFLW ASSERTED?
15346 NAD/TEST.FIFO.OVFLW.1
15347 -----
15348 190:
15349 =0*
15350 TEST.FIFO.OVFLW.1:
15351
15352 NAD/XFER.REQ ;OVFLW ASSERTED
15353 -----
15354 192:
15355 =1*
15356
15357 NAD/TEST.FIFO.OVFLW.0 ;OVFLW NOT ASSERTED...LOOP
15358 -----
15359
15360 U OE4E, 3792,15 MOV LS[#6] TO WR[0] ;NOP TO WAIT FOR IDC
15361 NOP.T16.1:
15362 DEC WR[0],
15363 DT(LONG)&SET.ALU.CC
15364 U OE4F, A100,35 JMP.IF[NEQ] TO [NOP.T16.1]
15365 U OE50, 88E4,F1
15366 U OE51, DB00,1A SKIP.IF[NO.FAST.INTERRUPT] ;SEE IF OVFLW L ASSERTED
15367 U OE52, 08E8,D4 JMP [ERROR.T16.1] ;OVFLW L ASSERTED WHEN FIFO
15368 ;ADRS WAS CLEAR
15369
15370 U OE53, 2F80,15 CLR WR[0] ;CALL CHECK.RESULT TO LOCK
15371 U OE54, 2F82,95 CLR WR[1] ;IN INTERMITTENT ERRORS
15372 U OE55, BE89,95 MOV WR[3] TO LS[ADDRESS.DATA]

```

22  
 :  
 :  
 U  
 U  
 U  
 U  
 U

E 10

ZZ-ENKCF-1.4 : ENKCF.MIC Test 16 FIFO B ADR15-MAR-83 Fiche 2 Frame E10 Sequence 327  
 : ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40 Page 321  
 : ENKCF.MIC Test 16 FIFO B ADRS CNTR OVFLW L BRANCH TEST 2 (M8388 IDC MODULE)

```

U OE56, 0869,3C :15373 JSR [CHECK.RESULT]
U OE57, 88E4,64 :15374 JMP [LOOP.T16.1]
 :15375
 :15376
U OE58, 3642,15 :15377 MOV LS[#2] TO WR[0] ;SET UP FOR ERROR 2
U OE59, BE82,15 :15378 MOV WR[0] TO LS[ERROR.NUMBER]
 :15379
 :15380
 :15381
U OE5A, 378C,15 :15382 MOV LS[#1FE] TO WR[0]
U OE5B, 2040,15 :15383 INC WR[0] ;ONE BEFORE OVFLW FOR AN R80
U OE5C, 3E10,15 :15384 MOV WR[0] TO LS[T8]
 :15385
U OE5D, 2F87,95 :15386 CLR WR[3] ;USE TO TRACK FIFO ADRS
 :15387
INCR.CNTR.T16.1:
U OE5E, 97A8,15 :15388 PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;WRITE A BYTE OF UNKNOWN DATA
 :15389 ; TO FIFO B TO INCR ADRS CNTR
U OE5F, 2047,95 :15390 INC WR[3] ;INCR ADRS TRACKER
U OE60, B788,15 :15391 MOV LS[#3] TO WR[0] ;NOP TO WAIT FOR IDC
 :15392
NOP.T16.2:
 :15393
U OE61, A100,35 :15394 DEC WR[0],
 :15395 DT(LONG)&SET.ALU.CC
U OE62, 88E6,11 :15396 JMP.IF[NEQ] TO [NOP.T16.2]
 :15397
 :15398
U OE63, DB00,1A :15399 SKIP.IF[NO.FAST.INTERRUPT]
U OE64, 08E8,D4 :15400 JMP [ERROR.T16.1] ;CHECK IF OVFLW L ASSERTED
 :15401 ;OVFLW L ASSERTED TOO SOON
 :15402
U OE65, 2F80,15 :15403 CLR WR[0] ;CALL CHECK.RESULT TO LOCK
U OE66, 2F82,95 :15404 CLR WR[1] ; IN INTERMITTENT ERRORS
U OE67, BE89,95 :15405 MOV WR[3] TO LS[ADDRESS.DATA]
U OE68, 0869,3C :15406 JSR [CHECK.RESULT]
U OE69, 88E4,64 :15407 JMP [LOOP.T16.1]
 :15408
 :15409
 :15410
U OE6A, 4E11,B5 :15411 CMP LS[T8] WITH WR[3], ;LAST COUNT?
 :15412 DT(LONG)&SET.ALU.CC
U OE6B, 88E5,E1 :15413 JMP.IF[NEQ] TO [INCR.CNTR.T16.1] ;JUMP IF NOT LAST COUNT
 :15414
U OE6C, B788,15 :15415 MOV LS[#3] TO WR[0]
U OE6D, BE82,15 :15416 MOV WR[0] TO LS[ERROR.NUMBER] ;SET UP FOR ERROR 3
 :15417
U OE6E, 97A8,15 :15418 PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;WRITE A BYTE OF DATA TO
 :15419 ;FIFO B
 :15420 ; TO INCR ADRS CNTR OVFLW
U OE6F, 2047,95 :15421 INC WR[3] ;INCR THE ADRS TRACKER
U OE70, B788,15 :15422 MOV LS[#3] TO WR[0] ;NOP TO WAIT FOR IDC
 :15423
NOP.T16.3:
 :15424
U OE71, A100,35 :15425 DEC WR[0],
 :15426 DT(LONG)&SET.ALU.CC
U OE72, 08E7,11 :15427 JMP.IF[NEQ] TO [NOP.T16.3]
 :15428
U OE73, DB00,1A :15429 SKIP.IF[NO.FAST.INTERRUPT]
U OE74, 88E7,64 :15430 JMP [OVFLW.OK.T16.1] ;OVFLW L ASSERTED?
 :15431 ;OK

```



```

U OE75, 08E8,D4 :15428 JMP [ERROR.T16.1] ;OVFLW DID NOT ASSERT
 :15429
 :15430 OVFLW.OK.T16.1:
 :15431
U OE76, 2F80,15 :15432 CLR WR[0] ;CALL CHECK.RESULT TO LOCK
U OE77, 2F82,95 :15433 CLR WR[1] ; IN INTERMITTENT ERRORS
U OE78, BE89,95 :15434 MOV WR[3] TO LS[ADDRESS.DATA]
U OE79, 0869,3C :15435 JSR [CHECK.RESULT]
U OE7A, 88E4,64 :15436 JMP [LOOP.T16.1]
 :15437
U OE7B, 3644,15 :15438 MOV LS[#4] TO WR[0]
U OE7C, BE82,15 :15439 MOV WR[0] TO LS[ERROR.NUMBER] ;SET UP FOR ERROR 4
 :15440 LOOP.T16.2:
U OE7D, 8872,EC :15441 JSR [DIAG.CODE] ;FORCE THE IDC TO DIAG.IDLE
U OE7E, 17C0,15 :15442 PORT.CONTROL [CLEAR.FIFO.CNTR] ;CLEAR FIFO B ADRS CNTR
 :15443 ; TO DEASSERT OVFLW L
 :15444
 :15445 ;SET UP THE CSR TO BRANCH TO
U OE7F, B706,15 :15446 MOV LS[SETCSR.F3] TO WR[0] ; THE CORRECT IDC ROUTINE THAT
 :15447 ; WILL USET OVFLW AND THEN
 :15448 ; BRANCH TO SEE IF OVFLW DID
 :15449 ; ASSERT
U OE80, AEC4,15 :15450 BIS WR[2] TO WR[0] ;SELECT AN R80
U OE81, 17A0,15 :15451 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
 :15452 ;MAINT=0, CRDY=0, F<2:0>=011
 :15453 -----
 :15454 -----
 :15455 *****
 :15456 * IDC *
 :15457 *****
 :15458 -----
 :15459 018:
 :15460 =000
 :15461 DIAG.IDLE:
 :15462
 :15463 BEN0/F0,
 :15464 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
 :15465 BEN2/F2,
 :15466 NAD/DIAG.IDLE
 :15467 -----
 :15468 01B:
 :15469 =011
 :15470 BEN0/CRDY.L,
 :15471 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
 :15472 NAD/TEST.FUNC3 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
 :15473 -----
 :15474 069:
 :15475 =0*1
 :15476 TEST.USET.OVFLW:
 :15477
 :15478 FUNC/SET.FIFO.OVF, ;USET OVFLW
 :15479 INCR.FIFO.CNTR/ON, ;CLK OVFLW
 :15480 NAD/TEST.FIFO.OVFLW.0
 :15481 -----
 :15482 068:

```

```

:15483 : =0*0
:15484 : TEST.FIFO.OVFLW.0:
:15485 :
:15486 : BEN1/FIFO.OVFLW.L, ;IS FIFO OVERFLOW ASSERTED?
:15487 : NAD/TEST.FIFO.OVFLW.1
:15488 : -----
:15489 : 190:
:15490 : =0*
:15491 : TEST.FIFO.OVFLW.1:
:15492 :
:15493 : NAD/XFER.REQ ;OVFLW ASSERTED
:15494 : -----
:15495 : 192:
:15496 : =1*
:15497 :
:15498 : NAD/TEST.FIFO.OVFLW.0 ;OVFLW NOT ASSERTED...LOOP
:15499 : -----
:15500 :
U OE82, 3792,15 :15501 : MOV LS[#6] TO WR[0] ;NOP TO WAIT FOR IDC
:15502 : NOP.T16.4:
:15503 : DEC WR[0],
:15504 : DT(LONG)&SET.ALU.CC
U OE83, A100,35 :15505 : JMP.IF[NEQ] TO [NOP.T16.4]
U OE84, 88E8,31 :15506 :
:15507 : SKIP.IF[NO.FAST.INTERRUPT] ;SEE IF OVFLW L ASSERTED
U OE85, DB00,1A :15508 : JMP [CALL.CH.RESULT.T16.1] ;OVFLW ASSERTED OK
:15509 :
:15510 : ERROR.T16.2:
U OE87, 3640,95 :15511 : MOV LS[#1] TO WR[1] ;OVFLW DID NOT ASSERT
U OE88, 2F80,15 :15512 : CLR WR[0] ;FORCE AN ERROR
U OE89, 6588,15 :15513 : CLR LS[ADDRESS.DATA] ;CLR FOR PRINTOUT
U OE8A, 0869,3C :15514 : JSR [CHECK.RESULT]
U OE8B, 08E7,D4 :15515 : JMP [LOOP.T16.2] ;LOOP HERE IF ERR & LOE IS SET
U OE8C, 88E9,74 :15516 : JMP [CLEANUP.T16.1] ;ELSE END TEST
:15517 :
:15518 :
:15519 : ERROR.T16.1:
U OE8D, 3640,95 :15520 : MOV LS[#1] TO WR[1] ;OVFLW L DID NOT ASSERT
U OE8E, 2F80,15 :15521 : CLR WR[0] ;FORCE AN ERROR
U OE8F, BE89,95 :15522 : MOV WR[3] TO LS[ADDRESS.DATA] ;PRINT OTHER DATA..LOOP CNT
U OE90, 0869,3C :15523 : JSR [CHECK.RESULT]
U OE91, 88E4,64 :15524 : JMP [LOOP.T16.1] ;LOOP HERE IF ERR & LOE IS SET
:15525 : ;ELSE SKIP OTHER ERRORS AND END
:15526 : CALL.CH.RESULT.T16.1:
:15527 :
U OE92, 2F80,15 :15528 : CLR WR[0] ;CALL CHECK.RESULT TO LOCK
U OE93, 2F82,95 :15529 : CLR WR[1] ;IN INTERMITTENT ERRORS
U OE94, 6588,15 :15530 : CLR LS[ADDRESS.DATA]
U OE95, 0869,3C :15531 : JSR [CHECK.RESULT]
U OE96, 08E7,D4 :15532 : JMP [LOOP.T16.2]
:15533 :
:15534 : CLEANUP.T16.1:
U OE97, 8875,3C :15535 : JSR [CLEANUP] ;CLEANUP THE CONDITIONS SET BY
:15536 : ;THIS TEST
:15537 : END.T16: ;END TEST

```

0000  
 0000  
 0000  
 0000

ZZ-ENKCF-1.4 : ENKCF.MIC Test 16 FIFO B ADR15-MAR-83 H 10 Fiche 2 Frame H10 Sequence 330  
: ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40 Page 324  
: ENKCF.MIC Test 16 FIFO B ADRS CNTR OVFLW L BRANCH TEST 2 (M8388 IDC MODULE)

:15538  
:15539  
:15540

ZZ  
:  
:

:15541 .PAGE "Test 17 LOOP COUNTER TEST (M8388 IDC MODULE)"

:15542 :++

:15543 :  
:15544 : Test Description:

:15545 :  
:15546 : This test is designed to test the loop counter and the associated  
:15547 : BENO/CNTR OVFLW branch condition. The loop counter will be loaded with  
:15548 : FF, which will assert CNTR OVFLW H. BENO/CNTR OVFLW H will be  
:15549 : selected and if the microcode branches correctly, the DAR will be  
:15550 : incremented.  
:15551 : The loop counter will then be loaded with a zero. The IDC microcode  
:15552 : will then execute an instruction which will select BENO/CNTR OVFLW H  
:15553 : and will increment the loop counter. When CNTR OVFLW asserts, the  
:15554 : microcode will branch to an instruction that will increment the DAR.  
:15555 : The CPU will read the DAR after every increment of the Loop Counter to  
:15556 : insure that the CNTR OVFLW asserts only at FF.

:15557 :  
:15558 : Assumptions:

:15559 :  
:15560 : It is assumed that the UINCR.DAR function works correctly.

:15561 :  
:15562 : Test Steps:

- :15563 :  
:15564 : 1. Set up error mask, error number and module number in LS  
:15565 : (for error printouts) and clear the previous error number in LS.  
:15566 : 2. Force the IDC microcode to DIAG.IDLE.  
:15567 : 3. Clear the DAR and set WR[2] = FF to indicate Loop count.  
:15568 : 4. Set up the CSR to branch to correct microcode routine.

:15569 :  
:15570 :  
:15571 : \*\*\*\*\*  
:15572 : \* IDC \*  
:15573 : \*\*\*\*\*

:15574 :  
:15575 : IDC1. Load the loop counter with a constant of FF.

- :15576 :  
:15577 : 5. Wait for a XFER.REQ from IDC.  
:15578 : 6. Clear CSR F<2:0> and send XFER GRANT to clear XFER.REQ.  
:15579 : 7. Set up the CSR to branch to correct microcode routine.

:15580 :  
:15581 :  
:15582 : \*\*\*\*\*  
:15583 : \* IDC \*  
:15584 : \*\*\*\*\*

- :15585 :  
:15586 : IDC2. Increment the loop counter.  
:15587 : IDC3. Select BENO/CNTR OVFLW H.  
:15588 : IDC4. If CNTR OVFLW H is asserted, then increment the DAR.  
:15589 : IDC5. Issue XFER REQ to flag the CPU.

- :15590 :  
:15591 : 8. Wait for a XFER.REQ from IDC.  
:15592 : 9. Clear CSR F<2:0> and send XFER GRANT to clear XFER.REQ.  
:15593 : 10. Check the DAR = 1.  
:15594 : 11. Increment the error number.  
:15595 : 12. Clear WR[2] to use as a counter.

```

:15596 : 13. Clear the DAR.
:15597 : 14. Set up the CSR to branch to the correct microcode routine.
:15598 :
:15599 : *****
:15600 : * IDC *
:15601 : *****
:15602 :
:15603 : IDC6. Load the loop counter with a constant of zero.
:15604 : IDC7. Send XFER REQ and go to DIAG.WAIT.
:15605 :
:15606 : 15. Wait for a XFER REQ from IDC.
:15607 : 16. Clear CSR F<2:0> and send XFER GRANT to clear the XFER REQ.
:15608 : 17. Set up the CSR to branch to the correct microcode routine.
:15609 :
:15610 : *****
:15611 : * IDC *
:15612 : *****
:15613 :
:15614 : IDC8. Increment the LOOP COUNTER.
:15615 : IDC9. Select BENO/CNTR OVFLW H.
:15616 : IDC10. If CNTR OVFLW H is asserted then increment the DAR.
:15617 : IDC11. Issue XFER REQ and go to DIAG.WAIT.
:15618 :
:15619 :
:15620 :
:15621 : 18. Wait for a XFER.REQ from the IDC.
:15622 : 19. Clear CSR F<2:0> and issue XFER GRANT to clear the XFER.REQ.
:15623 : 20. Check DAR = 0.
:15624 : 21. Clear the DAR.
:15625 : 22. Add 1 to WR[2].
:15626 : 23. While WR[2] < FF, perform Steps 17 - 23.
:15627 : 24. Increment the error number.
:15628 : 25. Perform Steps 14 - 16.
:15629 : 26. Check the DAR = 1. (LOOP COUNTER should overflow at FF + 1)
:15630 : 27. Clear IDC.
:15631 : 28. End test.

```

Errors:

Printout:

EXPECTED  
DAR

RECEIVED  
DAR

OTHER DATA  
WR[2]  
LOOP COUNT

ERROR 1 - The Loop Counter did not overflow when loaded with FF and incremented once.

ERROR 2 - The Loop Counter overflowed too soon.

ERROR 3 - The Loop Counter did not overflow when incremented to FF + 1.

Debug:

ERROR 1:

```

:15651 : The microcode will execute an a LOAD.LOOP.CNTR.[OFF] instruction
:15652 : which will assert ULOAD LPCNTR L. The loop counter is loaded with
:15653 : the constant of FF which comes from the microword bits
:15654 : <63:56>. This will assert CNTR OVFLW H.
:15655 : When the BEN0/CNTR OVFLW H IDC microinstruction is executed:
:15656 : BEN0 S0 H = L
:15657 : BEN0 S1 H = H
:15658 : BEN0 S2 H = L
:15659 :
:15660 : This will select CNTR OVFLW H as NUA 0 H. CNTR OVFLW H will be asserted
:15661 : so the IDC microcode will branch to an instruction that will
:15662 : increment the DAR.
:15663 :
:15664 : ERROR 2:
:15665 :
:15666 : The microcode will execute an a LOAD.LOOP.CNTR.[00] instruction
:15667 : which will assert ULOAD LPCNTR L. The loop counter is loaded with
:15668 : the constant of 00 which comes from the microword bits
:15669 : <63:56>.
:15670 : When BEN0/CNTR OVFLW H IDC microinstruction is executed:
:15671 : BEN0 S0 H = L
:15672 : BEN0 S1 H = H
:15673 : BEN0 S2 H = L
:15674 :
:15675 : This will select CNTR OVFLW H as NUA 0 H. CNTR OVFLW H will not
:15676 : be asserted so the IDC microcode will branch to an instruction
:15677 : that will send XFER REQ without incrementing the DAR.
:15678 :
:15679 : ERROR 3:
:15680 :
:15681 : When the Loop Counter is incremented to FF + 1, CNTR OVFLW H will
:15682 : assert. BEN0/CNTR OVFLW will be selected. The microcode will branch
:15683 : to an instruction which will increment the DAR and then issue
:15684 : XFER.REQ.
:15685 : --
:15686 : *****
:15687 : T.17:
U OE98, B65E,15 :15688 MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT 15 IN WR[0] FOR
:15689 : ; CONTROL AND STATUS WORD
U OE99, 3E80,15 :15690 MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND
:15691 : ; STATUS WORD - BIT 15
:15692 : ; INDICATES START OF TEST TO
:15693 : ; CONSOLE
U OE9A, 10E0,15 :15694 MISC [SET.CP.ATTN] ;ISSUE CPU ATTN TO CONSOLE
:15695 : WAIT.T17.0: ;
U OE9B, 88E9,B4 :15696 JMP [WAIT.T17.0] ;LOOP HERE WAITING FOR
:15697 : ; CONSOLE TO RESPOND
:15698 :
U OE9C, 65A0,15 :15699 CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
:15700 : ; IN LS
:15701 :
U OE9D, B64A,15 :15702 MOV LS[IDC] TO WR[0] ;STORE MODULE CODE IN LS TO
U OE9E, 3E8C,15 :15703 MOV WR[0] TO LS[MODULE.NUM] ; INDICATE IDC MODULE
:15704 :
U OE9F, B670,15 :15705 MOV LS[OTHER.DATA] TO WR[0] ;SET UP THE PRINT OTHER DATA

```

```

ZZ-ENKCF-1.4 ; ENKCF.MIC Test 17 LOOP COUNT15-MAR-83 L 10
: ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module
: ENKCF.MIC Test 17 LOOP COUNTER TEST (M8388 IDC MODULE)
Sequence 334 Page 328

U OEA0, 3E80,15 :15706 MOV WR[0] TO LS[CONTROL.STATUS] ; IF ERROR
:15707
U OEA1, B640,15 :15708 MOV LS[#1] TO WR[0] ;SET WRO = 1
U OEA2, BE82,15 :15709 MOV WR[0] TO LS[ERROR.NUMBER] ;SET UP ERROR NUMBER = 1 IN LS
:15710
U OEA3, 3776,15 :15711 MOV LS[DAR.MASK] TO WR[0] ;SET UP THE MASK FOR THE DAR
U OEA4, 3E8A,15 :15712 MOV WR[0] TO LS[ERROR.MASK]
:15713
U OEA5, 8872,EC :15714 JSR [DIAG.CODE] ;FORCE THE IDC TO DIAG.IDLE
:15715
U OEA6, 3627,15 :15716 MOV LS[FFF] TO WR[2] ;LOOP COUNT
:15717
:15718 LOOP.T17.1:
U OEA7, 2F80,15 :15719 CLR WR[0] ;WR[0] = 0
U OEA8, 97A4,15 :15720 PORT.WRITE PORT.ADRS[DAR] WITH WR[0] ;CLEAR THE DAR
:15721
U OEA9, B720,95 :15722 MOV LS[SETCSR.MAINT] TO WR[1] ; THE CORRECT IDC ROUTINE
U OEAA, 370E,15 :15723 MOV LS[SETCSR.F7] TO WR[0]
U OEAB, AEC2,15 :15724 BIS WR[1] TO WR[0]
U OEAC, 17A0,15 :15725 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:15726 ;MAINT=1,CRDY=0,F<2:0>=111
:15727 ;LOAD LPCNTR = FF
:15728 -----
:15729 -----
:15730 *****
:15731 * IDC *
:15732 *****
:15733 -----
:15734 :018:
:15735 =000
:15736 DIAG.IDLE:
:15737
:15738 BEN0/F0,
:15739 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:15740 BEN2/F2,
:15741 NAD/DIAG.IDLE
:15742 -----
:15743 :01F:
:15744 =111
:15745
:15746 BEN0/CRDY.L,
:15747 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:15748 NAD/TEST.FUNC7 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:15749 -----
:15750 :07D:
:15751 =1*1
:15752 TEST.F7:
:15753
:15754 FUNC/SET.ONLINE, ;SET ONLINE THE DRIVE SELECTED BY
:15755 ; DRIVE SEL <1:0> (FOR ONLINE TESTS)
:15756
:15757 LOAD.LOOP.CNTR.[OFF], ;LOAD LOOP CNTR = FF (FOR LP CNTR TST)
:15758
:15759 NAD/XFER.REQ ;DO NOT INCR DAR BETWEEN HERE
:15760 ; AND DIAG.IDLE

```

ZZ-  
:  
:  
:

U  
U  
  
U

ZZ-ENKCF-1.4  
: ENKCF.MCR  
: ENKCF.MIC

ENKCF.MIC

Test 17 LOOP COUNT15-MAR-83  
MICRO2 1M(01) 15-MAR-83 11:46:22  
Test 17 LOOP COUNTER TEST (M8388 IDC MODULE)

M 10

Fiche 2 Frame M10

Sequence 335  
Rev 01.40 Page 329

```
:15761 :-----
:15762 :1D8:
:15763 :XFER.REQ:
:15764 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:15765 : NAD/DIAG.WAIT
:15766 :-----
:15767 :1D7:
:15768 :DIAG.WAIT:
:15769 :
:15770 : BEN"/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:15771 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:15772 :-----
:15773 :159:
:15774 :=*0*
:15775 :DIAG.WAIT1:
:15776 :
:15777 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:15778 : ; COMMAND FROM THE CPU
:15779 :-----
:15780 :15B:
:15781 :=*1*
:15782 :DIAG.WAIT2:
:15783 :
:15784 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:15785 : ; WAIT SOME MORE
:15786 :-----
:15787 :WAIT.IDC.T17.1:
:15788 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR THE IDC
:15789 : JMP [IDC.DONE.T17.1] ;IDC DONE
:15790 : JMP [WAIT.IDC.T17.1] ;WAIT SOME MORE
:15791 :
:15792 :IDC.DONE.T17.1:
:15793 : MOV LS[SETCSR.F0] TO WR[0] ;CLEAR THE FUNCTION BITS
:15794 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:15795 : MISC2 [TRANSFER.GRANT] ;SEND XFER GRANT TO CLR REQ
:15796 :
:15797 : INC WR[2] ;INCREMENT THE TEST LP CNTR
:15798 :
:15799 : MOV LS[SETCSR.CRDY] TO WR[1]
:15800 : MOV LS[SETCSR.F4] TO WR[0] ;SET UP THE CSR TO BRANCH TO
:15801 : BIS WR[1] TO WR[0] ; THE CORRECT ROUTINE
:15802 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0]
:15803 : ;MAINT=0,CRDY=1,F<2:0>=100
:15804 : INCR THE LOOP CNTR
:15805 :-----
:15806 :-----
:15807 : *****
:15808 : * IDC *
:15809 : *****
:15810 :-----
:15811 :018:
:15812 :=000
:15813 :DIAG.IDLE:
:15814 :
:15815 : BEN0/F0,
```



ZZ-ENKCF-1.4  
: ENKCF.MCR  
: ENKCF.MIC

ENKCF.MIC

Test 17 LOOP COUNT15-MAR-83  
MICRO2 1M(01) 15-MAR-83 11:46:22  
Test 17 LOOP COUNTER TEST (M8388 IDC MODULE)

N 10

Fiche 2 Frame N10

Sequence 336  
Rev 01.40 Page 330

ZZ-  
:  
:

```
:15816 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:15817 : BEN2/F2,
:15818 : NAD/DIAG.IDLE
:15819 :-----
:15820 : 01C:
:15821 : =100
:15822 :
:15823 : BEN0/CRDY.L,
:15824 : BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:15825 : NAD/TEST.FUNC4 ;DEPENDENT ON CSR <CRDY> AND <MAINT>
:15826 :-----
:15827 : 06A:
:15828 : =0*0
:15829 : TEST.FUNC4:
:15830 :
:15831 : TEST.INCR.LPCNTR:
:15832 :
:15833 : INCR.LPCNTR/ON, ;INCREMENT THE LOOP CNTR
:15834 : BEN0/CNTR.OVFLW, ;DID LOOP CNTR OVFLW?
:15835 : NAD/BEN0.1
:15836 :-----
:15837 : 18A:
:15838 : =0
:15839 : BEN0.1:
:15840 :
:15841 : NAD/XFER.REQ ;SELECTED BRANCH 0 IS NOT ASSERTED
:15842 : ;FLAG THE CPU
:15843 :-----
:15844 : 18B:
:15845 : =1
:15846 : BEN0.2:
:15847 :
:15848 : INCR.DAR/ON, ;INCREMENT THE DAR
:15849 : NAD/XFER.REQ ;SELECTED BRANCH 0 WAS ASSERTED
:15850 : ;FLAG THE CPU
:15851 :-----
:15852 : 1D8:
:15853 : XFER.REQ:
:15854 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:15855 : NAD/DIAG.WAIT
:15856 :-----
:15857 : 1D7:
:15858 : DIAG.WAIT:
:15859 :
:15860 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:15861 : NAD/DIAG.WAIT1 ;BEFORE RETURNING TO DIAG.IDLE
:15862 :-----
:15863 : 159:
:15864 : =*0*
:15865 : DIAG.WAIT1:
:15866 :
:15867 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:15868 : ;COMMAND FROM THE CPU
:15869 :-----
:15870 : 15B:
```

[illegible]

```

:15926 : =110
:15927 :
:15928 : BEN0/CRDY.L,
:15929 : BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:15930 : NAD/TEST.FUNC6 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:15931 : -----
:15932 : 077:
:15933 : =1*1
:15934 : TEST.F6:
:15935 : ;DO NOT INCR DAR BETWEEN HERE AND
:15936 : ; DIAG.IDLE OR DIAG.WAIT
:15937 :
:15938 : LOAD.LOOP.CNTR.[00], ;LOAD LOOP CNTR WITH 00
:15939 :
:15940 : UCMD/SET.DLT, ;SET DLT FOR DLT TEST
:15941 :
:15942 : FUNC/SET.INT, ;SET INT FOR INTERRUPT TEST
:15943 :
:15944 : NAD/XFER.REQ ;SEND XFER REQ TO FLAG THE CPU
:15945 : -----
:15946 : 1D8:
:15947 : XFER.REQ:
:15948 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:15949 : NAD/DIAG.WAIT
:15950 : -----
:15951 : 1D7:
:15952 : DIAG.WAIT:
:15953 :
:15954 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:15955 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:15956 : -----
:15957 : 159:
:15958 : =*0*
:15959 : DIAG.WAIT1:
:15960 :
:15961 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:15962 : ; COMMAND FROM THE CPU
:15963 : -----
:15964 : 158:
:15965 : =*1*
:15966 : DIAG.WAIT2:
:15967 :
:15968 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:15969 : ; WAIT SOME MORE
:15970 : -----
:15971 : WAIT.IDC.T17.3:
:15972 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR THE IDC
:15973 : JMP [IDC.DONE.T17.3] ;IDC DONE
:15974 : JMP [WAIT.IDC.T17.3] ;WAIT SOME MORE
:15975 :
:15976 : IDC.DONE.T17.3:
:15977 : CLR WR[3] ;SET THE EXPECTED DATA
:15978 : INCR.CNTR.T17:
:15979 : MOV LS[SETCSR.F0] TO WR[0] ;CLR THE FUNCTION BITS
:15980 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR

```

```

U OED4, 90FA,15 :15981 MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR XFER REQ
 :15982
U OED5, 3712,95 :15983 MOV LS[SETCSR.CRDY] TO WR[1]
U OED6, 3708,15 :15984 MOV LS[SETCSR.F4] TO WR[0] ;SET UP THE CSR TO BRANCH TO
U OED7, AEC2,15 :15985 BIS WR[1] TO WR[0] ; THE CORRECT IDC ROUTINE
U OED8, 17A0,15 :15986 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
 :15987 ;MAINT=0,CRDY=1,F<2:0>=100
 :15988 ;INCR THE LOOP CNTR
 :15989 -----
 :15990 -----
 :15991 *****
 :15992 * IDC *
 :15993 *****
 :15994 -----
 :15995 018:
 :15996 =000
 :15997 DIAG.IDLE:
 :15998
 :15999 BEN0/F0,
 :16000 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
 :16001 BEN2/F2,
 :16002 NAD/DIAG.IDLE
 :16003 -----
 :16004 01C:
 :16005 =100
 :16006
 :16007 BEN0/CRDY.L,
 :16008 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
 :16009 NAD/TEST.FUNC4 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
 :16010 -----
 :16011 06A:
 :16012 =0*0
 :16013 TEST.FUNC4:
 :16014
 :16015 TEST.INCR.LPCNTR:
 :16016
 :16017 INCR.LPCNTR/ON, ;INCREMENT THE LOOP CNTR
 :16018 BEN0/CNTR.OVFLW, ;DID LOOP CNTR OVFLW?
 :16019 NAD/BEN0.1
 :16020 -----
 :16021 18A:
 :16022 =0
 :16023 BEN0.1:
 :16024
 :16025 NAD/XFER.REQ ;SELECTED BRANCH 0 IS NOT ASSERTED
 :16026 ; FLAG THE CPU
 :16027 -----
 :16028 18B:
 :16029 =1
 :16030 BEN0.2:
 :16031
 :16032 INCR.DAR/ON, ;INCREMENT THE DAR
 :16033 NAD/XFER.REQ ;SELECTED BRANCH 0 WAS ASSERTED
 :16034 ; FLAG THE CPU
 :16035 -----

```

```

:16036 :1D8:
:16037 :XFER.REQ:
:16038 :UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:16039 :NAD/DIAG.WAIT
:16040 -----
:16041 :1D7:
:16042 :DIAG.WAIT:
:16043
:16044 :BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:16045 :NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:16046 -----
:16047 :159:
:16048 :=*0*
:16049 :DIAG.WAIT1:
:16050
:16051 :NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:16052 :COMMAND FROM THE CPU
:16053 -----
:16054 :15B:
:16055 :=*1*
:16056 :DIAG.WAIT2:
:16057
:16058 :NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:16059 :WAIT SOME MORE
:16060 -----
:16061 WAIT.IDC.T17.4:
:16062 SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR THE IDC
:16063 JMP [IDC.DONE.T17.4] ;IDC DONE
:16064 JMP [WAIT.IDC.T17.4] ;WAIT SOME MORE
:16065
:16066 IDC.DONE.T17.4:
:16067 MOV LS[SETCSR.F0] TO WR[0] ;CLR THE FUNCTION BITS
:16068 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:16069 MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR XFER REQ
:16070
:16071 READ.DAR.T17.2:
:16072 MISC [SET.PORT.I.0] ;SELECT THE IDC
:16073 PORT.READ PORT.ADRS[DAR] ;SET UP TO READ THE DAR
:16074 MOV PORT TO LS[PORT0] ;READ THE DATA FROM DAR AND
:16075 ;SAVE IT IN LS
:16076 MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
:16077 MOV WR[3] TO WR[1] ;SET UP THE EXPECTED DATA
:16078 MOV WR[2] TO LS[ADDRESS.DATA] ;PRINT OTHER DATA..LOOP CNT
:16079 JSR [CHECK.RESULT] ;CHECK THE DATA
:16080 JMP [LOOP.T17.2] ;LOOP HERE IF ERR & LOE IS SET
:16081
:16082 CLR WR[0] ;WR[0] = 0
:16083 PORT.WRITE PORT.ADRS[DAR] WITH WR[0] ;CLR THE DAR
:16084 INC WR[2] ;INCR THE CPU LOOP CNTR
:16085 CMP WR[3] WITH LS[#1],
:16086 DT(LONG)&SET.ALU.CC ;TEST DONE?
:16087 JMP.IF[EQL] TO [CLEANUP.T17.1] ;JMP IF DONE
:16088
:16089 CMP WR[2] WITH LS[#FF],
:16090 DT(LONG)&SET.ALU.CC ;LAST COUNT BEFORE OVFLW ?

```

```

ZZ-ENKCF-1.4 ; ENKCF.MIC Test 17 LOOP COUNT15-MAR-83 F 11
; ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module
; ENKCF.MIC Test 17 LOOP COUNTER TEST (M8388 IDC MODULE)
Sequence 341 Page 335

U OEED, 08ED,21 ;16091 JMP.IF[NEQ] TO [INCR.CNTR.T17] ;NOT LAST COUNT
;16092
U OEEE, FF82,15 ;16093 INC LS[ERROR.NUMBER] ;SET UP FOR NEXT ERROR
U OEED, 2045,15 ;16094 INC WR[2] ;INCR THE TEST COUNTER
U OEFO, B641,95 ;16095 MOV LS[#1] TO WR[3] ;SET UP THE EXPECTED DATA
U OEF1, 08ED,24 ;16096 JMP [INCR.CNTR.T17] ;CHECK CNTR OVFLW
;16097
;16098 LOOP.T17.2:
;16099
U OEF2, 2F85,15 ;16100 CLR WR[2] ;CLR THE TEST COUNTER
U OEF3, 2F80,15 ;16101 CLR WR[0] ;WR[0] = 0
U OEF4, 97A4,15 ;16102 PORT.WRITE PORT.ADRS[DAR] WITH WR[0] ;CLR THE DAR
;16103
U OEF5, B720,95 ;16104 MOV LS[SETCSR.MAINT] TO WR[1] ;SET UP THE CSR TO BRANCH TO
U OEF6, B70C,15 ;16105 MOV LS[SETCSR.F6] TO WR[0] ;THE CORRECT IDC ROUTINE
U OEF7, AEC2,15 ;16106 BIS WR[1] TO WR[0] ;THAT WILL LOAD LP CNTR = 0
U OEF8, 17A0,15 ;16107 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
;16108 ;MAINT=1,CRDY=0,F<2:0>=110
;16109
;16110 ;LOAD THE LOOP CNTR = 00
;16111 -----
;16112 -----
;16113 *****
;16114 * IDC *
;16115 *****
;16116 -----
;16117 018:
;16118 =000
;16119 DIAG.IDLE:
;16120
;16121 BEN0/F0,
;16122 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
;16123 BEN2/F2,
;16124 NAD/DIAG.IDLE
;16125 -----
;16126 01E:
;16127 =110
;16128
;16129 BEN0/CRDY.L,
;16130 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
;16131 NAD/TEST.FUNC6 ;DEPENDENT ON CSR <CRDY> AND <MAINT>
;16132 -----
;16133 077:
;16134 =1*1
;16135 TEST.F6:
;16136 ;DO NOT INCR DAR BETWEEN HERE AND
;16137 ;DIAG.IDLE OR DIAG.WAIT
;16138
;16139 LOAD.LOOP.CNTR.[00], ;LOAD LOOP CNTR WITH 00
;16140
;16141 UCMD/SET.DLT, ;SET DLT FOR DLT TEST
;16142
;16143 FUNC/SET.INT, ;SET INT FOR INTERRUPT TEST
;16144
;16145 NAD/XFER.REQ ;SEND XFER REQ TO FLAG THE CPU

```

```

16146 :-----
16147 :1D8:
16148 :XFER.REQ:
16149 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
16150 : NAD/DIAG.WAIT
16151 :-----
16152 :1D7:
16153 :DIAG.WAIT:
16154 :
16155 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
16156 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
16157 :-----
16158 :159:
16159 :=*0*
16160 :DIAG.WAIT1:
16161 :
16162 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
16163 : ; COMMAND FROM THE CPU
16164 :-----
16165 :158:
16166 :=*1*
16167 :DIAG.WAIT2:
16168 :
16169 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
16170 : ; WAIT SOME MORE
16171 :-----
16172 :WAIT.IDC.T17.5:
16173 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR THE IDC
16174 : JMP [IDC.DONE.T17.5] ;IDC DONE
16175 : JMP [WAIT.IDC.T17.5] ;WAIT SOME MORE
16176 :-----
16177 :IDC.DONE.T17.5:
16178 : MOV LS[SETCSR.F0] TO WR[0] ;CLR THE FUNCTION BITS
16179 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
16180 : MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR XFER REQ
16181 :-----
16182 :ERR.INCR.LPCNTR.T17:
16183 : MOV LS[SETCSR.CRDY] TO WR[1]
16184 : MOV LS[SETCSR.F4] TO WR[0] ;SET UP THE CSR TO BRANCH TO
16185 : BIS WR[1] TO WR[0] ; THE CORRECT IDC ROUTINE
16186 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
16187 : ;MAINT=0,CRDY=1,F<2:0>=100
16188 : INCR THE LOOP CNTR
16189 :-----
16190 :-----
16191 : *****
16192 : * IDC *
16193 : *****
16194 :-----
16195 :018:
16196 :=000
16197 :DIAG.IDLE:
16198 :
16199 : BEN0/F0,
16200 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS

```

```

:16201 : BEN2/F2,
:16202 : NAD/DIAG.IDLE
:16203 :-----
:16204 : 01C:
:16205 : =100
:16206 :
:16207 : BEN0/CRDY.L,
:16208 : BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:16209 : NAD/TEST.FUNC4 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:16210 :-----
:16211 : 06A:
:16212 : =0*0
:16213 : TEST.FUNC4:
:16214 :
:16215 : TEST.INCR.LPCNTR:
:16216 :
:16217 : INCR.LPCNTR/ON, ;INCREMENT THE LOOP CNTR
:16218 : BEN0/CNTR.OVFLW, ;DID LOOP CNTR OVFLW?
:16219 : NAD/BEN0.1
:16220 :-----
:16221 : 18A:
:16222 : =0
:16223 : BEN0.1:
:16224 :
:16225 : NAD/XFER.REQ ;SELECTED BRANCH 0 IS NOT ASSERTED
:16226 : ; FLAG THE CPU
:16227 :-----
:16228 : 18B:
:16229 : =1
:16230 : BEN0.2:
:16231 :
:16232 : INCR.DAR/ON, ;INCREMENT THE DAR
:16233 : NAD/XFER.REQ ;SELECTED BRANCH 0 WAS ASSERTED
:16234 : ; FLAG THE CPU
:16235 :-----
:16236 : 1D8:
:16237 : XFER.REQ:
:16238 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:16239 : NAD/DIAG.WAIT
:16240 :-----
:16241 : 1D7:
:16242 : DIAG.WAIT:
:16243 :
:16244 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:16245 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:16246 :-----
:16247 : 159:
:16248 : =*0*
:16249 : DIAG.WAIT1:
:16250 :
:16251 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:16252 : ; COMMAND FROM THE CPU
:16253 :-----
:16254 : 15B:
:16255 : =*1*

```



```

:16256 :DIAG.WAIT2:
:16257 :
:16258 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:16259 : ; WAIT SOME MORE
:16260 :-----
:16261 WAIT.IDC.T17.6:
U OF03, DB00,1A :16262 SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR THE IDC
U OF04, 88F0,64 :16263 JMP [IDC.DONE.T17.6] ;IDC DONE
U OF05, 88F0,34 :16264 JMP [WAIT.IDC.T17.6] ;WAIT SOME MORE
:16265
:16266 IDC.DONE.T17.6:
U OF06, B700,15 :16267 MOV LS[SETCSR.F0] TO WR[0] ;CLR THE FUNCTION BITS
U OF07, 17A0,15 :16268 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
U OF08, 90FA,15 :16269 MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR XFER REQ
:16270
U OF09, 9090,15 :16271 MISC [SET.PORT.I.0] ;SELECT THE IDC
U OF0A, 1784,15 :16272 PORT.READ PORT.ADRS[DAR] ;SET UP TO READ THE DAR
U OF0B, 3B32,15 :16273 MOV PORT TO LS[PORT0] ;READ THE DATA FROM DAR AND
:16274 ; SAVE IT IN LS
U OF0C, 3732,15 :16275 MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
U OF0D, 2006,95 :16276 MOV WR[3] TO WR[1] ;SET UP THE EXPECTED DATA
U OF0E, 3E89,15 :16277 MOV WR[2] TO LS[ADDRESS.DATA] ;PRINT OTHER DATA..LOOP CNT
U OF0F, 0869,3C :16278 JSR [CHECK.RESULT] ;CHECK THE DATA
U OF10, 88F1,24 :16279 JMP [ERROR.T17.2] ;LOOP HERE IF ERR & LOE IS SET
U OF11, 08F1,C4 :16280 JMP [CLEANUP.T17.1] ;IF CL LOOP...END TEST
:16281
:16282 ERROR.T17.2:
U OF12, 2045,15 :16283 INC WR[2] ;INC THE TEST COUNTER
:16284
:16285 CMP WR[3] WITH LS[#1],
U OF13, CF41,B5 :16286 DT(LONG)&SET.ALU.CC
U OF14, 08F1,89 :16287 JMP.IF[EQL] TO [ERROR3.T17] ;ERROR 2 OR ERROR 3 LOOP?
:16288
:16289 CMP WR[2] WITH LS[#100],
U OF15, CF51,35 :16290 DT(LONG)&SET.ALU.CC
U OF16, 08EF,F1 :16291 JMP.IF[NEQ] TO [ERR.INCR.LPCNTR.T17] ;LAST COUNT?
U OF17, 88EF,24 :16292 JMP [LOOP.T17.2] ;GO INCR THE LP CNTR
:16293
:16294 ERROR3.T17:
U OF18, B7BA,95 :16295 MOV LS[#101] TO WR[1]
:16296 CMP WR[0] WITH WR[1],
U OF19, 2640,B5 :16297 DT(LONG)&SET.ALU.CC
U OF1A, 08EF,F1 :16298 JMP.IF[NEQ] TO [ERR.INCR.LPCNTR.T17] ;OVERFLOW??
U OF1B, 88EF,24 :16299 JMP [LOOP.T17.2] ;GO INC COUNTER
:16300
:16301 CLEANUP.T17.1:
U OF1C, 8875,3C :16302 JSR [CLEANUP] ;CLEANUP CONDITIONS SET BY THIS
:16303 ; TEST
:16304
:16305 END.T17:
:16306
:16307
:16308
:16309
:16310

```



:16312 :PAGE "Test 18 R80 BRANCH TEST (M8388 IDC MODULE)"

:16313 :++

:16314 :  
:16315 :Test Description:

:16316 :  
:16317 :This test will verify that the R80 branch condition operates correctly  
:16318 :when unasserted, and if an R80 is available on the system, the R80  
:16319 :branch condition will also be verified when asserted.

:16320 :  
:16321 :Assumptions:

:16322 :  
:16323 :It is assumed that all previous tests have run successfully.

:16324 :  
:16325 :Test Steps:

- :16326 :  
:16327 :1. Set up error mask, error number and module number in LS  
:16328 : (for error printouts) and clear the previous error number in LS.  
:16329 :2. If the DRIVE SIZING TEST has not previously been run then,  
:16330 : JSR to DRIVE.SIZE.  
:16331 :3. Force the IDC microcode to DIAG.IDLE.  
:16332 :4. Select an RL02 disk drive or a line with no disk attached.  
:16333 :5. Clear the DAR.  
:16334 :6. Set up the CSR to branch to the correct IDC routine.

:16335 :  
:16336 :  
:16337 :\*\*\*\*\*  
:16338 : \* IDC \*  
:16339 :\*\*\*\*\*

- :16340 :  
:16341 :IDC1. Select BEN0/R80.  
:16342 :IDC2. If R80 H is asserted, then increment the DAR.  
:16343 :IDC3. Send XFER REQ to flag the CPU.  
:16344 :IDC4. Go to DIAG.IDLE.

- :16345 :  
:16346 :7. Wait for an XFER REQ from IDC.  
:16347 :8. Clear CSR F<2:0> and send XFER GRANT to clear the XFER REQ.  
:16348 :9. If the DAR = 1 then issue error. (DAR will increment only if R80 was  
:16349 : asserted)  
:16350 :10. If an R80 is not available on the system the go to end of test.  
:16351 :11. Increment the error number.  
:16352 :12. Select an R80.  
:16353 :13. Clear the DAR.  
:16354 :14. Set up the CSR to branch to the correct IDC routine.

:16355 :  
:16356 :  
:16357 :\*\*\*\*\*  
:16358 : \* IDC \*  
:16359 :\*\*\*\*\*

:16360 :  
:16361 :IDC5. Perform steps IDC1-IDC4.

- :16362 :  
:16363 :15. Wait for an XFER REQ from IDC.  
:16364 :16. Clear CSR F<2:0> and send XFER GRANT to clear the XFER REQ.  
:16365 :17. If DAR = 0 then issue error.(DAR will increment if R80 is asserted).  
:16366 :18. CLEAR IDC.

16367 : 19. End test.

16368 :

16369 :

16370 :

16371 :

16372 :

16373 :

16374 :

16375 :

16376 :

16377 :

16378 :

16379 :

16380 :

16381 :

16382 :

16383 :

16384 :

16385 :

16386 :

16387 :

16388 :

16389 :

16390 :

16391 :

16392 :

16393 :

16394 :

16395 :

16396 :

16397 :

16398 :

16399 :

16400 :

16401 :

16402 :

16403 :

16404 :

16405 :

16406 :

16407 :

16408 :

16409 :

16410 :

16411 :

16412 :

16413 :

16414 :

16415 :

16416 :

16417 :

16418 :

16419 :

16420 :

16421 :

Errors:

Printout:

EXPECTED

RECEIVED

DAR

DAR

ERROR 1 - R80 H branch failed unasserted.

ERROR 2 - R80 H branch failed asserted.

Debug:

ERROR 1:

The DAR will be cleared.

A drive that asserts L RL02 H at the CSR SELECTED STATUS LATCH PAL is selected. This signal is inverted to R80 H = L. The IDC microcode will branch to an instruction that will select BEN0/R80. This outputs at the microsequencer will be:

BEN0 S2 H = H

BEN0 S1 H = L

BEN0 S0 H = L

This will select R80 H = L as NUA 0 H. The microcode will branch to an instruction will send XFER REQ to flag the CPU but will NOT increment the DAR.

The CPU will then read and check the DAR.

ERROR 2:

If no R80 is available, go to end of test.

The DAR is cleared.

A drive that does NOT assert LRL20 H at the CSR SELECTED STATUS LATCH PAL will be selected. This signal is inverted to become R80 H = H.

microcode will branch to an instruction that will select BEN0/R80. This outputs at the microsequencer will be:

BEN0 S2 H = H

BEN0 S1 H = L

BEN0 S0 H = L

This will select R80 H = H as NUA 0 H. The microcode will branch to an instruction that will increment the DAR and then send XFER REQ to flag the CPU.

The CPU will then read and check the DAR.

--

\*\*\*\*\*

T.18:

MOV LS[BEGIN.TEST] TO WR[0]

;SET BIT 15 IN WR[0] FOR

; CONTROL AND STATUS WORD

MOV WR[0] TO LS[CONTROL.STATUS]

;SET BIT 15 IN CONTROL AND

; STATUS WORD - BIT 15

; INDICATES START OF TEST TO

; CONSOLE

MISC [SET.CP.ATTN]

;ISSUE CPU ATTN TO CONSOLE

WAIT.T18.0:

U OF1D, B65E,15

U OF1E, 3E80,15

U OF1F, 10E0,15

U (

U (

U (

U (

U (

```

M 11
ZZ-ENKCF-1.4 ; ENKCF.MIC Test 18 R80 BRANCH15-MAR-83 Fiche 2 Frame M11 Sequence 348
: ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40 Page 342
: ENKCF.MIC Test 18 R80 BRANCH TEST (M8388 IDC MODULE)

U OF20, 08F2,04 :16422 JMP [WAIT.T18.0] ;LOOP HERE WAITING FOR
:16423 ; CONSOLE TO RESPOND
:16424
U OF21, 65A0,15 :16425 CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
:16426 ; IN LS
:16427
U OF22, B64A,15 :16428 MOV LS[IDC] TO WR[0] ;STORE MODULE CODE IN LS TO
U OF23, 3E8C,15 :16429 MOV WR[0] TO LS[MODULE.NUM] ; INDICATE IDC MODULE
:16430
:16431
U OF24, B640,15 :16432 MOV LS[#1] TO WR[0] ;SET WRO = 1
U OF25, BE82,15 :16433 MOV WR[0] TO LS[ERROR.NUMBER] ;SET UP ERROR NUMBER = 1 IN LS
:16434
U OF26, 3776,15 :16435 MOV LS[DAR.MASK] TO WR[0] ;GET THE MASK TO CHECK THE DAR
U OF27, 3E8A,15 :16436 MOV WR[0] TO LS[ERROR.MASK]
:16437
U OF28, B78E,15 :16438 MOV LS[DRIVE.STATUS] TO WR[0] ;HAS THE DRIVE SIZING TEST BEEN
:16439 BIT LS[BIT7] WITH WR[0], ; RUN?
U OF29, D94E,35 :16440 DT(LONG)&SET.ALU.CC
U OF2A, 08F2,C1 :16441 JMP.IF[BIT.SET] TO [SIZE.OK.T18.1] ;IF YES THEN GO RUN TEST
:16442
U OF2B, 8870,0C :16443 JSR [DRIVE.SIZE] ;GO RUN THE SIZING ROUTINE
:16444
:16445 SIZE.OK.T18.1:
:16446
U OF2C, 8872,EC :16447 JSR [DIAG.CODE] ;FORCE THE IDC TO DIAG.IDLE
:16448
:16449 ;SELECT AN RLO2
:16450
U OF2D, B78E,15 :16451 MOV LS[DRIVE.STATUS] TO WR[0]
:16452 BIT LS[BIT0] WITH WR[0], ;IS DRIVE 0 AN R80?
U OF2E, 5940,35 :16453 DT(LONG)&SET.ALU.CC
U OF2F, 88F3,29 :16454 JMP.IF[BITS.CLR] TO [SET.DS0.T18.1] ;JMP IF DRV 0 IS NOT AN R80
U OF30, B717,15 :16455 MOV LS[SETCSR.DS1] TO WR[2] ;SAVE THE DATA TO SEL DRIVE 1
U OF31, 88F3,34 :16456 JMP [TEST.T18.1]
:16457 SET.DS0.T18.1:
U OF32, 3715,15 :16458 MOV LS[SETCSR.DS0] TO WR[2] ;SAVE THE DATA TO SEL DRIVE 0
:16459
:16460 TEST.T18.1:
:16461 LOOP.T18.1:
U OF33, 2F80,15 :16462 CLR WR[0] ;WR[0] = 0
U OF34, 97A4,15 :16463 PORT.WRITE PORT.ADRS[DAR] WITH WR[0] ;CLEAR THE DAR
U OF35, B712,15 :16464 MOV LS[SETCSR.CRDY] TO WR[0] ;SET UP THE CSR TO BRANCH TO
U OF36, B720,95 :16465 MOV LS[SETCSR.MAINT] TO WR[1] ; BRANCH TO THE CORRECT IDC
U OF37, AEC2,15 :16466 BIS WR[1] TO WR[0] ; ROUTINE
U OF38, B708,95 :16467 MOV LS[SETCSR.F4] TO WR[1]
U OF39, AEC2,15 :16468 BIS WR[1] TO WR[0]
U OF3A, AEC4,15 :16469 BIS WR[2] TO WR[0] ;SELECT AN RLO2
U OF3B, 17A0,15 :16470 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:16471 ;MAINT=1,CRDY=1,F<2:0>=100
:16472 -----
:16473 -----
:16474 :
:16475 : *****
:16476 : * IDC *
: : *****

```

```

:16477 :-----
:16478 :018:
:16479 :=000
:16480 :DIAG.IDLE:
:16481 :
:16482 : BEN0/F0,
:16483 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:16484 : BEN2/F2,
:16485 : NAD/DIAG.IDLE
:16486 :-----
:16487 :01C:
:16488 :=100
:16489 :
:16490 : BEN0/CRDY.L,
:16491 : BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:16492 : NAD/TEST.FUNC4 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:16493 :-----
:16494 :06E:
:16495 :=1*0
:16496 :TEST.R80.1:
:16497 :
:16498 : BEN0/R80, ;IS R80 ASSERTED CSR <26>?
:16499 : BEN2/R80.FORMAT.H, ;IS CSR BIT <29> ASSERTED?
:16500 : NAD/R80.BR.TST
:16501 :-----
:16502 :07A:
:16503 :=0*0
:16504 :R80.BR.TST:
:16505 :
:16506 : NAD/XFER.REQ ;R80 NOT ASSERTED AND R80 FORMAT
:16507 : ; NOT ASSERTED DO NOT INC THE DAR
:16508 :-----
:16509 :07B:
:16510 :=0*1
:16511 :R80.BR.TST.1:
:16512 :
:16513 : INCR.DAR/ON, ;INC THE DAR..R80 ASSERTED
:16514 : NAD/XFER.REQ ;FLAG THE CPU
:16515 :-----
:16516 :07E:
:16517 :=1*0
:16518 :R80.BR.TST.2:
:16519 :
:16520 : INCR.DAR/ON, ;INC THE DAR..R80.FORMAT ASSERTED
:16521 : NAD/XFER.REQ ;FLAG THE CPU
:16522 :-----
:16523 :=
:16524 :END THE BRANCH BLOCK
:16525 :-----
:16526 :1D8:
:16527 :XFER.REQ:
:16528 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:16529 : NAD/DIAG.WAIT
:16530 :-----
:16531 :1D7:

```

```

:16532 :DIAG.WAIT:
:16533 :
:16534 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:16535 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:16536 :-----
:16537 :159:
:16538 :=*0*
:16539 :DIAG.WAIT1:
:16540 :
:16541 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:16542 : ; COMMAND FROM THE CPU
:16543 :-----
:16544 :15B:
:16545 :=*1*
:16546 :DIAG.WAIT2:
:16547 :
:16548 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:16549 : ; WAIT SOME MORE
:16550 :-----
:16551 :WAIT.IDC.T18.1:
U OF3C, DB00,1A :16552 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR THE IDC
U OF3D, 88F3,F4 :16553 : JMP [IDC.DONE.T18.1] ;IDC DONE
U OF3E, 88F3,C4 :16554 : JMP [WAIT.IDC.T18.1] ;WAIT SOME MORE
:16555 :
:16556 :IDC.DONE.T18.1:
U OF3F, B700,15 :16557 : MOV LS[SETCSR.F0] TO WR[0] ;CLEAR THE FUNCTION BITS
U OF40, 17A0,15 :16558 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
U OF41, 90FA,15 :16559 : MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR XFER REQ
:16560 : ; AND FORCE TO DIAG.IDLE
:16561 :READ.DAR.T18.1:
U OF42, 9090,15 :16562 : MISC [SET.PORT.1.0] ;SELECT THE IDC
U OF43, 1784,15 :16563 : PORT.READ PORT.ADRS[DAR] ;SET UP TO READ THE DAR
U OF44, 3B32,15 :16564 : MOV PORT TO LS[PORT0] ;READ THE DATA FROM DAR AND
:16565 : ; SAVE IT IN LS
U OF45, 3732,15 :16566 : MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
U OF46, 2F82,95 :16567 : CLR WR[1] ;SET UP THE EXPECTED DATA
U OF47, 0869,3C :16568 : JSR [CHECK.RESULT] ;CHECK THE DATA
U OF48, 88F3,34 :16569 : JMP [LOOP.T18.1] ;LOOP HERE IF ERR & LOE IS SET
:16570 :
:16571 :
:16572 : ;SELECT AN R80 DRIVE IF ONE IS AVAILABLE...ELSE SKIP REST OF TEST
:16573 :
U OF49, B78E,15 :16574 : MOV LS[DRIVE.STATUS] TO WR[0]
U OF4A, 3790,95 :16575 : MOV LS[FFFFFFFF0] TO WR[1]
U OF4B, AF42,15 :16576 : BIC WR[1] TO WR[0] ;CHECK BITS <3:0>
:16577 : CMP LS[ZERO] WITH WR[0], ;ANY R80S?
:16578 : DT(LONG)&SET.ALU.CC
U OF4C, CE9C,35 :16579 : JMP.IF[NEQ] TO [FIND.R80.T18] ;YES..GO DO TEST
:16580 :
U OF4E, 08F7,54 :16581 : JMP [CLEANUP.T18.1] ;NO R80, CLEANUP AND END TEST
:16582 :
:16583 :FIND.R80.T18:
U OF4F, B78E,15 :16584 : MOV LS[DRIVE.STATUS] TO WR[0]
:16585 : BIT LS[BIT0] WITH WR[0],
U OF50, 5940,35 :16586 : DT(LONG)&SET.ALU.CC ;IS DRIVE 0 AN R80?

```

```

U OF51, 08F5,71 :16587 JMP.IF[BIT.SET] TO [SEL.DR0.T18.1] ;JMP IF DRV 0 = R80
 :16588
 :16589 BIT LS[BIT1] WITH WR[0], ;IS DRIVE 1 AN R80?
U OF52, D942,35 :16590 DT(LONG)&SET.ALU.CC
U OF53, 88F5,91 :16591 JMP.IF[BIT.SET] TO [SEL.DR1.T18.1] ;JMP IF DRV 1 = R80
 :16592
 :16593 BIT LS[BIT2] WITH WR[0], ;IS DRIVE 2 AN R80?
U OF54, D944,35 :16594 DT(LONG)&SET.ALU.CC
U OF55, 08F5,B1 :16595 JMP.IF[BIT.SET] TO [SEL.DR2.T18.1] \ ;JMP IF DRV 2 = R80
 :16596
U OF56, 08F5,D4 :16597 JMP [SEL.DR3.T18.1] ;JMP AND SEL DR 3
 :16598
 :16599 SEL.DR0.T18.1:
U OF57, 3715,15 :16600 MOV LS[SETCSR.DS0] TO WR[2] ;SAV DRV SEL = 0
U OF58, 08F5,E4 :16601 JMP [TEST.T18.2] ;GO DO TEST
 :16602
 :16603 SEL.DR1.T18.1:
U OF59, B717,15 :16604 MOV LS[SETCSR.DS1] TO WR[2] ;SAV DRV SEL = 1
U OF5A, 08F5,E4 :16605 JMP [TEST.T18.2] ;GO DO TEST
 :16606
 :16607 SEL.DR2.T18.1:
U OF5B, 3719,15 :16608 MOV LS[SETCSR.DS2] TO WR[2] ;SAV DRV SEL = 2
U OF5C, 08F5,E4 :16609 JMP [TEST.T18.2] ;GO DO TEST
 :16610
 :16611 SEL.DR3.T18.1:
U OF5D, B71B,15 :16612 MOV LS[SETCSR.DS3] TO WR[2] ;SAV DRV SEL = 3
 :16613
 :16614 TEST.T18.2:
U OF5E, FF82,15 :16615 INC LS[ERROR.NUMBER] ;SET UP FOR ERROR 2
 :16616
 :16617 LOOP.T18.2:
U OF5F, 2F80,15 :16617 CLR WR[0] ;WR[0] = 0
U OF60, 97A4,15 :16618 PORT.WRITE PORT.ADRS[DAR] WITH WR[0] ;CLEAR THE DAR
U OF61, B712,15 :16619 MOV LS[SETCSR.CRDY] TO WR[0] ;SET UP THE CSR TO BRANCH TO
U OF62, B720,95 :16620 MOV LS[SETCSR.MAINT] TO WR[1] ; BRANCH TO THE CORRECT IDC
U OF63, AEC2,15 :16621 BIS WR[1] TO WR[0] ; ROUTINE
U OF64, B708,95 :16622 MOV LS[SETCSR.F4] TO WR[1]
U OF65, AEC2,15 :16623 BIS WR[1] TO WR[0]
U OF66, AEC4,15 :16624 BIS WR[2] TO WR[0] ;SELECT AN R80
U OF67, 17A0,15 :16625 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
 :16626
 :16627 -----
 :16628 -----
 :16629 *****
 :16630 * IDC *
 :16631 *****
 :16632 -----
 :16633 018:
 :16634 =000
 :16635 DIAG.IDLE:
 :16636
 :16637 BEN0/F0,
 :16638 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
 :16639 BEN2/F2,
 :16640 NAD/DIAG.IDLE
 :16641 -----

```



```

:16642 :01C:
:16643 :=100
:16644 :
:16645 : BEN0/CRDY.L,
:16646 : BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:16647 : NAD/TEST.FUNC4 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:16648 :-----
:16649 :06E:
:16650 :=1*0
:16651 :TEST.R80.1
:16652 :
:16653 : BEN0/R80, ;IS R80 ASSERTED CSR <26>?
:16654 : BEN2/R80.FORMAT.H, ;IS CSR BIT <29> ASSERTED?
:16655 : NAD/R80.BR.TST
:16656 :-----
:16657 :07A:
:16658 :=0*0
:16659 :R80.BR.TST:
:16660 :
:16661 : NAD/XFER.REQ ;R80 NOT ASSERTED AND R80 FORMAT
:16662 : ; NOT ASSERTED DO NOT INC THE DAR
:16663 :-----
:16664 :07B:
:16665 :=0*1
:16666 :R80.BR.TST.1:
:16667 :
:16668 : INCR.DAR/ON, ;INC THE DAR..R80 ASSERTED
:16669 : NAD/XFER.REQ ;FLAG THE CPU
:16670 :-----
:16671 :07E:
:16672 :=1*0
:16673 :R80.BR.TST.2:
:16674 :
:16675 : INCR.DAR/ON, ;INC THE DAR..R80.FORMAT ASSERTED
:16676 : NAD/XFER.REQ ;FLAG THE CPU
:16677 :-----
:16678 :=
:16679 :END THE BRANCH BLOCK
:16680 :-----
:16681 :1D8:
:16682 :XFER.REQ:
:16683 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:16684 : NAD/DIAG.WAIT
:16685 :-----
:16686 :1D7:
:16687 :DIAG.WAIT:
:16688 :
:16689 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:16690 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:16691 :-----
:16692 :159:
:16693 :=*J*
:16694 :DIAG.WAIT1:
:16695 :
:16696 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND

```

[illegible]

```
:16735 .PAGE "Test 19 R80 FORMAT BRANCH TEST (M8388 IDC MODULE)"
:16736 :++
:16737 : Test Description:
:16738 :
:16739 : This test is designed to insure that the BEN2/R80 FORMAT branch
:16740 : condition operates correctly. CSR bit <29> (R80 FORMAT) will be set
:16741 : by writing to the CSR. The IDC will then select BEN2/R80 FORMAT.
:16742 : If R80 FORMAT is asserted, then DAR will be incremented.
:16743 : If R80 FORMAT is NOT asserted the DAR will not be incremented.
:16744 : The CPU will then read and check the DAR = 1. CSR bit <29> will then
:16745 : be cleared. The IDC will again select BEN2/R80 FORMAT.
:16746 : If R80 FORMAT is NOT asserted the DAR will not be incremented.
:16747 : If R80 FORMAT is asserted, then DAR will be incremented.
:16748 : The CPU will then read and check the DAR = 0.
:16749 :
:16750 : Assumptions:
:16751 :
:16752 : It is assumed that the CONTROL STATUS REGISTER TEST has been
:16753 : previously run successfully.
:16754 :
:16755 : Test Steps:
:16756 :
:16757 : 1. Set up error mask, error number and module number in LS
:16758 : (for error printouts) and clear the previous error number in LS.
:16759 : 2. If the DRIVE SIZING TEST has not previously been run then,
:16760 : JSR to DRIVE.SIZE.
:16761 : 3. Force the IDC microcode to DIAG.IDLE.
:16762 : 4. Select a drive which asserts L RLO2 H.
:16763 : 5. Clear the DAR.
:16764 : 6. Set CSR <29> R80 FORMAT.
:16765 : 8. Set up the CSR to branch to the correct IDC routine.
:16766 :
:16767 : *****
:16768 : * IDC *
:16769 : *****
:16770 :
:16771 : IDC1. Select BEN2/R80 FORMAT.
:16772 : IDC2. Increment the DAR if R80 FORMAT is asserted.
:16773 : IDC3. Send XFER REQ to flag the CPU.
:16774 :
:16775 : 8. Wait for a XFER REQ from IDC.
:16776 : 9. Clear CSR F<2:0> and send XFER GRANT to clear the XFER REQ.
:16777 : 10. Check the DAR = 1.
:16778 : 11. Increment the error number.
:16779 : 12. Clear the DAR.
:16780 : 13. Clear CSR <29> R80 FORMAT.
:16781 : 14. Set up the CSR to branch to the correct IDC routine.
:16782 :
:16783 : *****
:16784 : * IDC *
:16785 : *****
:16786 :
:16787 : IDC4. Select BEN2/R80 FORMAT.
:16788 : IDC5. Increment the DAR if R80 FORMAT is asserted.
:16789 : IDC6. Send XFER REQ to flag the CPU.
```

:16790 :  
:16791 : 15. Wait for a XFER REQ from IDC.  
:16792 : 16. Clear CSR F<2:0> and send XFER GRANT to clear the XFER REQ.  
:16793 : 17. Check the DAR = 0.  
:16794 : 18. Clear IDC.  
:16795 : 19. End test.  
:16796 :  
:16797 :  
:16798 :  
:16799 :  
:16800 :  
:16801 :  
:16802 :  
:16803 :  
:16804 :  
:16805 :  
:16806 :  
:16807 :  
:16808 :  
:16809 :  
:16810 :  
:16811 :  
:16812 :  
:16813 :  
:16814 :  
:16815 :  
:16816 :  
:16817 :  
:16818 :  
:16819 :  
:16820 :  
:16821 :  
:16822 :  
:16823 :  
:16824 :  
:16825 :  
:16826 :  
:16827 :  
:16828 :  
:16829 :  
:16830 :  
:16831 :  
:16832 :  
:16833 :  
:16834 :  
:16835 :  
:16836 :  
:16837 :  
:16838 :  
:16839 :  
:16840 :  
:16841 :  
:16842 :  
:16843 :  
:16844 :

Errors:

Printout:

EXPECTED  
DAR

RECEIVED  
DAR

ERROR 1 - BEN2/R80 FORMAT branch failed when R80 FORMAT was asserted.

ERROR 2 - BEN2/R80 FORMAT branch failed when R80 FORMAT was  
NOT asserted.

Debug:

Since CSR <29> (R80 FORMAT) has previously been tested, it is most  
likely that the branch condition has failed.

ERROR 1:

Select a drive which asserts L RL02 H so that the inverse of that,  
R80 H = L.  
Set CSR <29>, R80 FORMAT H.  
The IDC microcode will branch to an BEN2/R80.FORMAT.H instruction.  
The outputs of the microsequencer will be:

BEN2 S2 H = H  
BEN2 S1 H = H  
BEN2 S0 H = L

This will select R80 FORMAT H = H as NUA 2 H. The IDC microcode  
will branch to an instruction that will increment the DAR and then  
send XFER REQ to flag the CPU.  
The CPU will then read and check the DAR.

ERROR 2:

Select a drive which asserts L RL02 H so that the inverse of that,  
R80 H = L.  
Clear CSR <29>, R80 FORMAT H.  
The IDC microcode will branch to an BEN2/R80.FORMAT.H instruction.  
The outputs of the microsequencer will be:

BEN2 S2 H = H  
BEN2 S1 H = H  
BEN2 S0 H = L

This will select R80 FORMAT L = H as NUA 2 H. The IDC microcode  
will branch to an instruction that will send XFER REQ to flag  
the CPU and will NOT increment the DAR.  
The CPU will then read and check the DAR.

T.19:

MOV LS[BEGIN.TEST] TO WR[0]

;SET BIT 15 IN WR[0] FOR

|                 |        |                                      |                                  |
|-----------------|--------|--------------------------------------|----------------------------------|
| U OF77, 3E80,15 | :16845 |                                      | : CONTROL AND STATUS WORD        |
|                 | :16846 | MOV WR[0] TO LS[CONTROL.STATUS]      | : SET BIT 15 IN CONTROL AND      |
|                 | :16847 |                                      | : STATUS WORD - BIT 15           |
|                 | :16848 |                                      | : INDICATES START OF TEST TO     |
|                 | :16849 |                                      | : CONSOLE                        |
| U OF78, 10E0,15 | :16850 | MISC [SET.CP.ATTN]                   | : ISSUE CPU ATTN TO CONSOLE      |
|                 | :16851 | WAIT.T19.0:                          |                                  |
| U OF79, 08F7,94 | :16852 | JMP [WAIT.T19.0]                     | : LOOP HERE WAITING FOR          |
|                 | :16853 |                                      | : CONSOLE TO RESPOND             |
|                 | :16854 |                                      |                                  |
| U OF7A, 65A0,15 | :16855 | CLR LS[PREVIOUS.ERROR]               | : CLEAR PREVIOUS ERROR NUMBER    |
|                 | :16856 |                                      | : IN LS                          |
|                 | :16857 |                                      |                                  |
| U OF7B, B64A,15 | :16858 | MOV LS[IDC] TO WR[0]                 | : STORE MODULE CODE IN LS TO     |
| U OF7C, 3E8C,15 | :16859 | MOV WR[0] TO LS[MODULE.NUM]          | : INDICATE IDC MODULE            |
|                 | :16860 |                                      |                                  |
|                 | :16861 |                                      |                                  |
| U OF7D, B640,15 | :16862 | MOV LS[#1] TO WR[0]                  | : SET WRO = 1                    |
| U OF7E, BE82,15 | :16863 | MOV WR[0] TO LS[ERROR.NUMBER]        | : SET UP ERROR NUMBER = 1 IN LS  |
|                 | :16864 |                                      |                                  |
| U OF7F, 3776,15 | :16865 | MOV LS[DAR.MASK] TO WR[0]            |                                  |
| U OF80, 3E8A,15 | :16866 | MOV WR[0] TO LS[ERROR.MASK]          | : SET MASK TO CHECK <18:0>       |
|                 | :16867 |                                      |                                  |
| U OF81, B78E,15 | :16868 | MOV LS[DRIVE.STATUS] TO WR[0]        | : HAS THE DRIVE SIZING TEST BEEN |
|                 | :16869 | BIT LS[BIT7] WITH WR[0],             | : RUN?                           |
| U OF82, D94E,35 | :16870 | DT(LONG)&SET.ALU.CC                  |                                  |
| U OF83, 08F8,51 | :16871 | JMP.IF[BIT.SET] TO [SIZE.OK.T19.1]   | : IF YES THEN GO RUN TEST        |
|                 | :16872 |                                      |                                  |
| U OF84, 8870,0C | :16873 | JSR [DRIVE.SIZE]                     | : GO RUN THE SIZING ROUTINE      |
|                 | :16874 |                                      |                                  |
|                 | :16875 | SIZE.OK.T19.1:                       |                                  |
|                 | :16876 |                                      |                                  |
| U OF85, 8872,EC | :16877 | JSR [DIAG.CODE]                      | : FORCE THE IDC TO DIAG.IDLE     |
|                 | :16878 |                                      |                                  |
|                 | :16879 | ;SELECT AN RL02                      |                                  |
|                 | :16880 |                                      |                                  |
| U OF86, B78E,15 | :16881 | MOV LS[DRIVE.STATUS] TO WR[0]        |                                  |
|                 | :16882 | BIT LS[BIT0] WITH WR[0],             | : IS DRIVE 0 AN R80?             |
| U OF87, 5940,35 | :16883 | DT(LONG)&SET.ALU.CC                  |                                  |
| U OF88, 08F8,B9 | :16884 | JMP.IF[BITS.CLR] TO [SET.DS0.T19.1]  | : JMP IF DRV 0 IS NOT AN R80     |
| U OF89, B717,15 | :16885 | MOV LS[SETCSR.DS1] TO WR[2]          | : SAVE THE DATA TO SEL DRIVE 1   |
| U OF8A, 08F8,C4 | :16886 | JMP [TEST.T19.1]                     |                                  |
|                 | :16887 | SET.DS0.T19.1:                       |                                  |
| U OF8B, 3715,15 | :16888 | MOV LS[SETCSR.DS0] TO WR[2]          | : SAVE THE DATA TO SEL DRIVE 0   |
|                 | :16889 |                                      |                                  |
|                 | :16890 | TEST.T19.1:                          |                                  |
|                 | :16891 | LOOP.T19.1:                          |                                  |
| U OF8C, 2F80,15 | :16892 | CLR WR[0]                            |                                  |
| U OF8D, 97A4,15 | :16893 | PORT.WRITE PORT.ADRS[DAR] WITH WR[0] | : CLEAR THE DAR                  |
|                 | :16894 |                                      |                                  |
| U OF8E, B712,15 | :16895 | MOV LS[SETCSR.CRDY] TO WR[0]         | : SET UP THE CSR TO BRANCH TO    |
| U OF8F, B720,95 | :16896 | MOV LS[SETCSR.MAINT] TO WR[1]        | : THE ROUTINE THAT WILL SELECT   |
| U OF90, AEC2,15 | :16897 | BIS WR[1] TO WR[0]                   | : BEN2/R80.FORMAT                |
| U OF91, B708,95 | :16898 | MOV LS[SETCSR.F4] TO WR[1]           |                                  |
| U OF92, AEC2,15 | :16899 | BIS WR[1] TO WR[0]                   |                                  |

```

U OF93, AEC4,15 :16900 BIS WR[2] TO WR[0] ;INCLUDE THE DRIVE SELECT
U OF94, C77A,15 :16901 BIS LS[BIT29] TO WR[0] ;SET CSR BIT <29> R80 FORMAT
U OF95, 17A0,15 :16902 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:16903 ;MAINT=1,CRDY=1,F<2:0>=100
:16904 -----
:16905 -----
:16906 *****
:16907 * IDC *
:16908 *****
:16909 -----
:16910 018:
:16911 =000
:16912 DIAG.IDLE:
:16913
:16914 BEN0/F0,
:16915 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:16916 BEN2/F2,
:16917 NAD/DIAG.IDLE
:16918 -----
:16919 01C:
:16920 =100
:16921
:16922 BEN0/CRDY.L,
:16923 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:16924 NAD/TEST.FUNC4 ;DEPENDENT ON CSR <CRDY> AND <MAINT>
:16925 -----
:16926 06E:
:16927 =1*0
:16928 TEST.R80.1:
:16929
:16930 BEN0/R80, ;IS R80 ASSERTED CSR <26>?
:16931 BEN2/R80.FORMAT.H, ;IS CSR BIT <29> ASSERTED?
:16932 NAD/R80.BR.TST
:16933 -----
:16934 07A:
:16935 =0*0
:16936 R80.BR.TST:
:16937
:16938 NAD/XFER.REQ ;R80 NOT ASSERTED AND R80 FORMAT
:16939 ;NOT ASSERTED DO NOT INC THE DAR
:16940 -----
:16941 07B:
:16942 =0*1
:16943 R80.BR.TST.1:
:16944
:16945 INCR.DAR/ON, ;INC THE DAR..R80 ASSERTED
:16946 NAD/XFER.REQ ;FLAG THE CPU
:16947 -----
:16948 07E:
:16949 =1*0
:16950 R80.BR.TST.2:
:16951
:16952 INCR.DAR/ON, ;INC THE DAR..R80.FORMAT ASSERTED
:16953 NAD/XFER.REQ ;FLAG THE CPU
:16954 -----

```

```

:16955 :=
:16956 :END THE BRANCH BLOCK
:16957 -----
:16958 :1D8:
:16959 :XFER.REQ:
:16960 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:16961 : NAD/DIAG.WAIT
:16962 -----
:16963 :1D7:
:16964 :DIAG.WAIT:
:16965 :
:16966 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:16967 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:16968 -----
:16969 :159:
:16970 :=*0*
:16971 :DIAG.WAIT1:
:16972 :
:16973 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:16974 : ; COMMAND FROM THE CPU
:16975 -----
:16976 :15B:
:16977 :=*1*
:16978 :DIAG.WAIT2:
:16979 :
:16980 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:16981 : ; WAIT SOME MORE
:16982 -----
:16983 :WAIT.IDC.T19.1:
:16984 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ FROM IDC
:16985 : JMP [IDC.DONE.T19.1] ;XFER REQ SET
:16986 : JMP [WAIT.IDC.T19.1] ;WAIT SOME MORE
:16987 -----
:16988 :IDC.DONE.T19.1:
:16989 : MOV LS[SETCSR.F0] TO WR[0] ;CLEAR CSR F<2:0>
:16990 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:16991 : MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
:16992 -----
:16993 :READ.DAR.T19.1:
:16994 : MISC [SET.PORT.I.0] ;SELECT THE IDC
:16995 : PORT.READ PORT.ADRS[DAR] ;SET UP TO READ THE DAR
:16996 : MOV PORT TO LS[PORT0] ;READ THE DATA FROM DAR AND
:16997 : ; SAVE IT IN LS
:16998 : MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
:16999 : MOV LS[#1] TO WR[1] ;SET UP THE EXPECTED DATA
:17000 : JSR [CHECK.RESULT] ;CHECK THE DATA
:17001 : JMP [LOOP.T19.1] ;LOOP HERE IF ERR & LOE IS SET
:17002 -----
:17003 : INC LS[ERROR.NUMBER] ;SET UP FOR ERROR 2
:17004 -----
:17005 :LOOP.T19.2:
:17006 : CLR WR[0]
:17007 : PORT.WRITE PORT.ADRS[DAR] WITH WR[0] ;CLEAR THE DAR
:17008 -----
:17009 : MOV LS[SETCSR.CRDY] TO WR[0] ;SET UP THE CSR TO BRANCH TO

```

U OF96, DB00,1A  
U OF97, 88F9,94  
U OF98, 88F9,64

U OF99, B700,15  
U OF9A, 17A0,15  
U OF9B, 90FA,15

U OF9C, 9090,15  
U OF9D, 1784,15  
U OF9E, 3B32,15

U OF9F, 3732,15  
U OFA0, 3640,95  
U OFA1, 0869,3C  
U OFA2, 08F8,C4

U OFA3, FF82,15

U OFA4, 2F80,15  
U OFA5, 97A4,15

U OFA6, B712,15

ZZ-ENKCF-1.4 ;  
: ENKCF.MCR  
: ENKCF.MIC

ENKCF.MIC

Test 19 R80 FORMAT15-MAR-83  
MICRO2 1M(01) 15-MAR-83 11:46:22  
Test 19 R80 FORMAT BRANCH TEST (M8388 IDC MODULE)

K 12  
Fiche 2 Frame K12

Sequence 359  
Rev 01.40 Page 353

ZZ-  
:  
:

```
U OFA7, B720,95 :17010 MOV LS[SETCSR.MAINT] TO WR[1] ; THE ROUTINE THAT WILL SELECT
U OFA8, AEC2,15 :17011 BIS WR[1] TO WR[0] ; BEN2/R80.FORMAT
U OFA9, B708,95 :17012 MOV LS[SETCSR.F4] TO WR[1]
U OFAA, AEC2,15 :17013 BIS WR[1] TO WR[0] ;BIT 29 - R80 FORMAT = 0
U OFAB, AEC4,15 :17014 BIS WR[2] TO WR[0] ;INCLUDE THE DRIVE SELECT
U OFAC, 17A0,15 :17015 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:17016 ;MAINT=1,CRDY=1,F<2:0>=100
:17017 -----
:17018 -----
:17019 *****
:17020 * IDC *
:17021 *****
:17022 -----
:17023 018:
:17024 =000
:17025 DIAG.IDLE:
:17026
:17027 BEN0/F0,
:17028 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:17029 BEN2/F2,
:17030 NAD/DIAG.IDLE
:17031 -----
:17032 01C:
:17033 =100
:17034
:17035 BEN0/CRDY.L,
:17036 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:17037 NAD/TEST.FUNC4 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:17038 -----
:17039 06E:
:17040 =1*0
:17041 TEST.R80.1:
:17042
:17043 BEN0/R80, ;IS R80 ASSERTED CSR <26>?
:17044 BEN2/R80.FORMAT.H, ;IS CSR BIT <29> ASSERTED?
:17045 NAD/R80.BR.TST
:17046 -----
:17047 07A:
:17048 =0*0
:17049 R80.BR.TST:
:17050
:17051 NAD/XFER.REQ ;R80 NOT ASSERTED AND R80 FORMAT
:17052 ; NOT ASSERTED DO NOT INC THE DAR
:17053 -----
:17054 07B:
:17055 =0*1
:17056 R80.BR.TST.1:
:17057
:17058 INCR.DAR/ON, ;INC THE DAR..R80 ASSERTED
:17059 NAD/XFER.REQ ;FLAG THE CPU
:17060 -----
:17061 07E:
:17062 =1*0
:17063 R80.BR.TST.2:
:17064
```



```

:17065 : INCR.DAR/ON, ;INC THE DAR..R80.FORMAT ASSERTED
:17066 : NAD/XFER.REQ ;FLAG THE CPU
:17067 :-----
:17068 : =
:17069 : END THE BRANCH BLOCK
:17070 :-----
:17071 : 1D8:
:17072 : XFER.REQ:
:17073 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:17074 : NAD/DIAG.WAIT
:17075 :-----
:17076 : 1D7:
:17077 : DIAG.WAIT:
:17078 :
:17079 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:17080 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:17081 :-----
:17082 : 159:
:17083 : =*0*
:17084 : DIAG.WAIT1:
:17085 :
:17086 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:17087 : ; COMMAND FROM THE CPU
:17088 :-----
:17089 : 15B:
:17090 : =*1*
:17091 : DIAG.WAIT2:
:17092 :
:17093 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:17094 : ; WAIT SOME MORE
:17095 :-----
:17096 : WAIT.IDC.T19.2:
:17097 : SKIP IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ FROM IDC
:17098 : JMP [IDC.DONE.T19.2] ;XFER REQ SET
:17099 : JMP [WAIT.IDC.T19.2] ;WAIT SOME MORE
:17100 :
:17101 : IDC.DONE.T19.2:
:17102 : MOV LS[SETCSR.F0] TO WR[0] ;CLEAR CSR F<2:0>
:17103 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:17104 : MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
:17105 :
:17106 : READ.DAR.T19.2:
:17107 : MISC [SET.PORT.I.0] ;SELECT THE IDC
:17108 : PORT.READ PORT.ADRS[DAR] ;SET UP TO READ THE DAR
:17109 : MOV PORT TO LS[PORT0] ;READ THE DATA FROM DAR AND
:17110 : ; SAVE IT IN LS
:17111 : MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
:17112 : CLR WR[1] ;SET UP THE EXPECTED DATA
:17113 : JSR [CHECK.RESULT] ;CHECK THE DATA
:17114 : JMP [LOOP.T19.2] ;LOOP HERE IF ERR & LOE IS SET
:17115 :
:17116 : JSR [CLEANUP] ;CLEANUP THE CONDITIONS SET BY
:17117 : ; THIS TEST
:17118 :
:17119 : END.T19: ;END TEST

```

U OFAD, DB00,1A  
 U OFAE, 08FB,04  
 U OFAF, 08FA,D4

U OFB0, B700,15  
 U OFB1, 17A0,15  
 U OFB2, 90FA,15

U OFB3, 9090,15  
 U OFB4, 1784,15  
 U OFB5, 3B32,15

U OFB6, 3732,15  
 U OFB7, 2F82,95  
 U OFB8, 0869,3C  
 U OFB9, 08FA,44

U OFBA, 8875,3C

U 0

U 0

U 0

U 0

U 0

U 0

U 0

U 0

U 0

U 0

U 0

U 0

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

ZZ-ENKCF-1.4 : ENKCF.MIC Test 19 R80 FORMAT15-MAR-83 M 12 Fiche 2 Frame M12 Sequence 361  
: ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40 Page 355  
: ENKCF.MIC Test 19 R80 FORMAT BRANCH TEST (M8388 IDC MODULE)

:17120  
:17121

ZZ-1  
:  
:  
U 1  
U 1  
U 1  
U 1  
U 1  
U 1

```

17122 PAGE "Test 1A AUTOMATIC SKIP SECTOR INHIBIT BRANCH TEST (M8388 IDC MODULE)"
17123 **
17124 Test Description:
17125
17126 This test will check the BEN2/ASSI H branch condition is working
17127 correctly. CSR <27> (ASSI) will be set. The IDC microcode will
17128 select BEN2/ASSI H. If ASSI H is asserted the DAR will be incremented
17129 twice. If ASSI H is not asserted the DAR will be incremented once.
17130 CSR <27> (ASSI) will then be cleared. The IDC microcode will again
17131 select BEN2/ASSI H. If ASSI H is NOT asserted, the DAR will be
17132 incremented once. If ASSI H is asserted the DAR will be
17133 incremented twice.
17134
17135 Assumptions:
17136
17137 It is assumed that the CSR test has previously been run successfully.
17138
17139 Test Steps:
17140
17141 1. Set up error mask, error number and module number in LS
17142 (for error printouts) and clear the previous error number in LS.
17143 2. Force the IDC to DIAG.IDLE.
17144 3. Clear the DAR.
17145 4. Set CSR <27> ASSI.
17146 5. Set up the CSR to branch to the correct IDC routine.
17147
17148 *****
17149 * IDC *
17150 *****
17151
17152 IDC1. Select BEN2/ASSI H.
17153 IDC2. If ASSI H is asserted, increment the DAR twice.
17154 If ASSI H is NOT asserted, increment the DAR once.
17155 IDC3. Send XFER REQ to flag the CPU.
17156
17157 6. Wait for XFER REQ from IDC.
17158 7. Clear CSR F<2:0> and send XFER GRANT to clear XFER REQ.
17159 8. Check the DAR = 2.
17160 9. Increment the error number.
17161 10. Clear the DAR.
17162 11. Clear CSR <27> ASSI.
17163 12. Set up the CSR to branch to the correct IDC routine.
17164
17165 *****
17166 * IDC *
17167 *****
17168
17169 IDC4. Perform Steps IDC1 - IDC3.
17170
17171 13. Wait for XFER REQ from IDC.
17172 14. Clear CSR F<2:0> and send XFER GRANT to clear XFER REQ.
17173 15. Check DAR = 1.
17174 16. Clear IDC.
17175 17. End test.
17176

```

U 1  
U 1  
U 1

U 1  
U 1  
U 1

U 1

U. 1  
U. 1

U 1

U 1  
U 1

U 1

U 1  
U 1

04

U 1  
U 1

U 1  
U 1

U 1  
U 1

```

:17177 : Errors:
:17178 :
:17179 :
:17180 : Printout:
:17181 : EXPECTED RECEIVED
:17182 : DAR DAR
:17183 :
:17184 : ERROR 1 - BEN2/ASSI H failed with ASSI H asserted.
:17185 :
:17186 : ERROR 2 - BEN2/ASSI H failed with ASSI H NOT asserted.
:17187 :
:17188 : Debug:
:17189 :
:17190 : Since CSR <27> (ASSI) has previously been tested, it is most
:17191 : likely that the branch condition has failed.
:17192 :
:17193 : ERROR 1:
:17194 : Set CSR <27>, ASSI H.
:17195 : The IDC microcode will branch to an BEN2/IASS instruction.
:17196 : The outputs of the microsequencer will be:
:17197 : BEN2 S2 H = H
:17198 : BEN2 S1 H = L
:17199 : BEN2 S0 H = H
:17200 : This will select ASSI H = H as NUA 2 H. The IDC microcode
:17201 : will branch to set of instructions that will increment the DAR
:17202 : twice and send XFER REQ to flag the CPU.
:17203 : The CPU will then read and check the DAR = 2.
:17204 :
:17205 : ERROR 2:
:17206 : Clear CSR <27>, ASSI H.
:17207 : The IDC microcode will branch to an BEN2/IASS instruction.
:17208 : The outputs of the microsequencer will be:
:17209 : BEN2 S2 H = H
:17210 : BEN2 S1 H = L
:17211 : BEN2 S0 H = H
:17212 : This will select ASSI = L as NUA 2 H. The IDC microcode
:17213 : will branch to an instruction that will increment the DAR
:17214 : and then send XFER REQ to flag the CPU.
:17215 : The CPU will then read and check the DAR = 1.
:17216 : --
:17217 : *****
:17218 : T.1A:
:17219 : MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT 15 IN WR[0] FOR
:17220 : ; CONTROL AND STATUS WORD
:17221 : MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND
:17222 : ; STATUS WORD - BIT 15
:17223 : ; INDICATES START OF TEST TO
:17224 : ; CONSOLE
:17225 : MISC [SET.CP.ATTN] ;ISSUE CPU ATTN TO CONSOLE
:17226 : WAIT.T1A.0:
:17227 : JMP [WAIT.T1A.0] ;LOOP HERE WAITING FOR
:17228 : ; CONSOLE TO RESPOND
:17229 :
:17230 : CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
:17231 : ; IN LS

```

```

U OFBB, B65E,15
U OFBC, 3E80,15
U OFBD, 10E0,15
U OFBE, 88FB,E4
U OFBF, 65A0,15

```

[illegible]

```

17287 :-----
17288 :182:
17289 :=0**
17290 :GROUP1:
17291 :
17292 : BEN2/R80.FORMAT.H, ;WHEN THE R80 FORMAT BIT SETS..BRANCH
17293 : NAD/GROUP1 ; TO NEXT WORD AND THEN BRANCH ON THE
17294 : ; FUNCTION BITS
17295 :-----
17296 :186:
17297 :=1**
17298 :
17299 : BEN0/F0,
17300 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
17301 : BEN2/F2,
17302 : NAD/GROUP1.F0
17303 :-----
17304 :02E:
17305 :=110
17306 :
17307 : BEN2/IASS, ;IS ASSI ASSERTED?
17308 : NAD/BEN2.1
17309 :-----
17310 :1A0:
17311 :=0**
17312 :BEN2.1:
17313 :
17314 : INCR.DAR/ON, ;INCR DAR
17315 : NAD/XFER.REQ ;FLAG THE CPU
17316 :-----
17317 :1A4:
17318 :=1**
17319 :BEN2.2:
17320 :
17321 : INCR.DAR/ON, ;INCR DAR IF BEN2 CONDITION ASSERTED
17322 : NAD/BEN2.1 ;INCR DAR AGAIN IF BEN2 COND ASSERTED
17323 :-----
17324 :1D8:
17325 :XFER.REQ:
17326 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
17327 : NAD/DIAG.WAIT
17328 :-----
17329 :1D7:
17330 :DIAG.WAIT:
17331 :
17332 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
17333 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
17334 :-----
17335 :159:
17336 :=*0*
17337 :DIAG.WAIT1:
17338 :
17339 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
17340 : ; COMMAND FROM THE CPU
17341 :-----

```

ZZ-  
 :  
 :  
 U 1  
 U 1  
 U 1  
 U 1  
 U 1

[illegible]

```

:17397 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:17398 : BEN2/F2,
:17399 : NAD/DIAG.IDLE
:17400 :-----
:17401 : 01F:
:17402 : =111
:17403 :
:17404 : BEN0/CRDY.L,
:17405 : BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:17406 : NAD/TEST.FUNC7 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:17407 :-----
:17408 : 079:
:17409 : =0*1
:17410 :
:17411 : NAD/GROUP1
:17412 :-----
:17413 : 182:
:17414 : =0**
:17415 : GROUP1:
:17416 :
:17417 : BEN2/R80.FORMAT.H, ;WHEN THE R80 FORMAT BIT SETS..BRANCH
:17418 : NAD/GROUP1 ; TO NEXT WORD AND THEN BRANCH ON THE
:17419 : ; FUNCTION BITS
:17420 :-----
:17421 : 186:
:17422 : =1**
:17423 :
:17424 : BEN0/F0,
:17425 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:17426 : BEN2/F2,
:17427 : NAD/GROUP1.F0
:17428 :-----
:17429 : 02E:
:17430 : =110
:17431 :
:17432 : BEN2/IASS, ;IS ASSI ASSERTED?
:17433 : NAD/BEN2.1
:17434 :-----
:17435 : 1A0:
:17436 : =0**
:17437 : BEN2.1:
:17438 :
:17439 : INCR.DAR/ON, ;INCR DAR
:17440 : NAD/XFER.REQ ;FLAG THE CPU
:17441 :-----
:17442 : 1A4:
:17443 : =1**
:17444 : BEN2.2:
:17445 :
:17446 : INCR.DAR/ON, ;INCR DAR IF BEN2 CONDITION ASSERTED
:17447 : NAD/BEN2.1 ;INCR DAR AGAIN IF BEN2 COND ASSERTED
:17448 :-----
:17449 : 1D8:
:17450 : XFER.REQ:
:17451 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU

```



MICRO2 1M(01)

15-MAR-83 11:46:22

ENKCF Micro-diagnostic for IDC module

Rev 01.40

Page 362

Test 1A AUTOMATIC SKIP SECTOR INHIBIT BRANCH TEST (M8388 IDC MODULE)

```

17452 : NAD/DIAG.WAIT
17453 :-----
17454 : 1D7:
17455 : DIAG.WAIT:
17456 :
17457 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
17458 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
17459 :-----
17460 : 159:
17461 : =*0*
17462 : DIAG.WAIT1:
17463 :
17464 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
17465 : ; COMMAND FROM THE CPU
17466 :-----
17467 : 15B:
17468 : =*1*
17469 : DIAG.WAIT2:
17470 :
17471 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
17472 : ; WAIT SOME MORE
17473 :-----
17474 : WAIT.IDC.T1A.2:
17475 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ FROM IDC
17476 : JMP [IDC.DONE.T1A.2] ;XFER REQ SET
17477 : JMP [WAIT.IDC.T1A.2] ;WAIT SOME MORE
17478 :
17479 : IDC.DONE.T1A.2:
17480 : MOV LS[SETCSR.F0] TO WR[0] ;CLEAR CSR F<2:0>
17481 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
17482 : MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
17483 :
17484 : READ.DAR.T1A.2:
17485 : MISC [SET.PORT.I.0] ;SELECT THE IDC
17486 : PORT.READ PORT.ADRS[DAR] ;SET UP TO READ THE DAR
17487 : MOV PORT TO LS[PORT0] ;READ THE DATA FROM DAR AND
17488 : ; SAVE IT IN LS
17489 : MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
17490 : MOV LS[#1] TO WR[1] ;SET UP EXPECTED DATA
17491 : JSR [CHECK.RESULT] ;CHECK THE DATA
17492 : JMP [LOOP.T1A.2] ;LOOP HERE IF ERR & LOE IS SET
17493 :
17494 : JSR [CLEANUP] ;CLEANUP CONDITIONS SET BY THIS
17495 : ; TEST
17496 :
17497 : END.T1A: ;END TEST
17498 :
17499 :
17500 :

```

U 1

U 1

01  
41

01

U 1  
U 1

01  
01

U 1

111

U 1

11 1

01

U 1

1111

101

107

# Ch

U 1

101

1 U 1

1011

1

101  
111

1

141

U 1

U T  
H 1

1001

107

101

101

1

```

:17501 .PAGE 'Test 1B ONLINE PAL & BRANCH TEST, DR SEL CSR<9:8> (M8388 IDC MODULE)'
:17502 .++
:17503
:17504 .Test Description:
:17505
:17506 .This test is designed to verify the ONLINE REGISTER PAL and the
:17507 .associated BEN1/ONLINE H branch condition. A portion of the DRIVE
:17508 .SELECT LOGIC PAL will also be verified in this test.
:17509 .When the BEN1/ONLINE H branch condition is selected, if ONLINE H
:17510 .is asserted the DAR will be incremented and then XFER.REQ will be
:17511 .issued. Otherwise XFER.REQ is issued.
:17512 .CSR Bits <9:8> will be used to select the current drive.
:17513
:17514 .All drives will be cleared online. Each one will then be checked
:17515 .to insure that it is clear.
:17516
:17517 .Drive 0 will then be selected as the current drive and set online.
:17518 .BEN1/ONLINE H will be selected and the CPU will be called to check the
:17519 .DAR=1. Each of the other drives will be selected and checked that they
:17520 .remained clear.
:17521
:17522 .Drive 0 will again be selected as the current drive and online will be
:17523 .cleared. BEN1/ONLINE H will be selected and the CPU will be called to
:17524 .check the DAR=0.
:17525
:17526 .The next drive will then be selected, and the same sequence of testing
:17527 .will be performed as was using Drive 0 until all drives <3:0> have
:17528 .been tested.
:17529
:17530 .Assumptions:
:17531
:17532 .It is assumed that the UINCR.DAR function works correctly.
:17533
:17534 .Test Steps:
:17535
:17536 .1. Set up error mask, error number and module number in LS
:17537 .(for error printouts) and clear the previous error number in LS.
:17538 .2. Force the IDC microcode to DIAG.IDLE.
:17539 .3. Clear the DAR.
:17540 .4. Move the number 0 to WR[2] to indicate the current drive selected.
:17541 .5. Move WR[2] to CSR <DS1> <DS0>.
:17542 .6. Set up the CSR to branch to the correct microcode routine.
:17543
:17544
:17545 .*****
:17546 .* IDC *
:17547 .*****
:17548
:17549 .IDC1. Set FUNC BITS <35:32> = 2, CLEAR ONLINE.
:17550 .IDC2. Issue XFER REQ to flag the CPU and go to DIAG.WAIT.
:17551
:17552 .7. Wait for a XFER.REQ from IDC.
:17553 .8. Clear CSR F<2:0> and issue XFER GRANT to clear XFER.REQ.
:17554 .9. Set WR[2] = to next drive number and repeat Steps 5 - 9 until all
:17555 .drives have been cleared.

```

```

17556 : 10. Set WR[2] = 0.
17557 : 11. Move WR[2] to CSR <DS1> <DS0>.
17558 : 12. Set up the CSR to branch to the correct microcode routine.
17559 :
17560 :
17561 : *****
17562 : * IDC *
17563 : *****
17564 :
17565 : IDC3. Select BEN1/ONLINE H.
17566 : IDC4. If ONLINE H is asserted, then increment the DAR.
17567 : IDC5. Issue XFER REQ to flag the CPU.
17568 :
17569 : 13. Wait for a XFER REQ from IDC.
17570 : 14. Clear CSR F<2:0> and send XFER GRANT to clear the XFER REQ.
17571 : 15. Check the DAR = 0.
17572 : 16. Move the next drive number to WR[2] and perform Steps 11 - 16
17573 : until all the drives have been checked.
17574 : 17. Increment the error number.
17575 : 18. Clear the DAR.
17576 : 19. Set WR[2] = 0.
17577 : 20. Move WR[2] to CSR <DS1><DS0>.
17578 : 21. Set up the CSR to branch to the correct microcode routine.
17579 :
17580 : *****
17581 : * IDC *
17582 : *****
17583 :
17584 : IDC6. Set FUNC BITS <35:32> = A, SET.ONLINE.
17585 :
17586 : 22. Wait for XFER.REQ from the IDC.
17587 : 23. Clear CSR F<2:0> and issue XFER GRANT to clear the XFER.REQ.
17588 : 24. Set up the CSR to branch to the correct IDC routine.
17589 :
17590 : *****
17591 : * IDC *
17592 : *****
17593 :
17594 : IDC7. Select BEN1/ONLINE H.
17595 : IDC8. If ONLINE H is asserted then increment the DAR.
17596 : IDC9. Issue XFER REQ to flag the CPU and go to DIAG.WAIT.
17597 :
17598 : 25. Wait for XFER.REQ from the IDC.
17599 : 26. Clear CSR F<2:0> and issue XFER GRANT to clear the XFER.REQ.
17600 : 27. Check the DAR = 1.
17601 : 28. Increment the error number.
17602 : 29. Check that all the other drives remained cleared.
17603 : 30. Increment the error number.
17604 : 31. Move WR[2] to CSR <DS1> <DS0>.
17605 : 32. Clear the DAR.
17606 : 33. Set up the CSR to branch to the correct microcode routine.
17607 :
17608 : *****
17609 : * IDC *
17610 : *****

```

[illegible]

:17611 :  
:17612 :  
:17613 :  
:17614 :  
:17615 :  
:17616 :  
:17617 :  
:17618 :  
:17619 :  
:17620 :  
:17621 :  
:17622 :  
:17623 :  
:17624 :  
:17625 :  
:17626 :  
:17627 :  
:17628 :  
:17629 :  
:17630 :  
:17631 :  
:17632 :  
:17633 :  
:17634 :  
:17635 :  
:17636 :  
:17637 :  
:17638 :  
:17639 :  
:17640 :  
:17641 :  
:17642 :  
:17643 :  
:17644 :  
:17645 :  
:17646 :  
:17647 :  
:17648 :  
:17649 :  
:17650 :  
:17651 :  
:17652 :  
:17653 :  
:17654 :  
:17655 :  
:17656 :  
:17657 :  
:17658 :  
:17659 :  
:17660 :  
:17661 :  
:17662 :  
:17663 :  
:17664 :  
:17665 :

IDC10. Set FUNC BITS <35:32> = 9, CLR.ONLINE.

34. Wait for a XFER REQ from IDC.  
35. Clear CSR F<2:0> and send XFER GRANT to clear the XFER REQ.  
36. Set up the CSR to branch to the correct microcode routine.

\*\*\*\*\*  
\* IDC \*  
\*\*\*\*\*

IDC11. Select BEN1/ONLINE H.  
IDC12. If ONLINE H is asserted, then increment the DAR.  
IDC13. Issue XFER REQ to flag the CPU.

37. Wait for a XFER REQ from IDC.  
38. Clear CSR F<2:0> and send XFER GRANT to clear the XFER REQ.  
39. Check the DAR = 0.  
40. Move the next drive number to WR[2] and perform Steps 20 - 40  
until all drives have been checked.  
41. Clear IDC.  
42. End test.

Errors:

Printout:

| EXPECTED<br>DAR | RECEIVED<br>DAR | OTHER DATA<br>Drive Number |
|-----------------|-----------------|----------------------------|
|-----------------|-----------------|----------------------------|

- ERROR 1 - The drive indicated by WR[2] was not cleared online.  
ERROR 2 - The drive number in OTHER DATA was not set online when  
selected by CSR<9:8> and USET.ONLINE was asserted.  
ERROR 3 - The drive number in OTHER DATA was set online when another  
drive number was selected.  
ERROR 4 - The drive number in OTHER DATA was not cleared online when  
selected by CSR <9:8> and CLR.ONLINE was asserted.

Debug:

ERROR 1:

The Drive Number which is being selected is put is CSR<9:8>. These  
bits are DS1 H AND DS0 H. These bits are inputs to the DR SEL LOG  
PAL and because USEL DRV H = L, these bits are output asDRIVE SEL 1  
and DRIVE SEL 0. DRIVE SEL <1:0> are inputs to the ONLINE REG PAL.  
CSR bits <9:8> are first written = [00] to select drive 0. Then the IDC  
microcode branches to a FUNC/CLR.ONLINE instruction. the outputs of the  
microsequencer will be:  
UFUNC 3 H = H

MICRO2 1M(01) 15-MAR-83 11:46:22

### ENKCF Micro-diagnostic for IDC module

Rev 01.40

Page 366

Test 1B ONLINE PAL & BRANCH TEST, DR SEL CSR<9:8> (M8388 IDC MODULE)

```

17666 : UFUNC 2 H = L
17667 : UFUNC 1 H = L
17668 : UFUNC 0 H = H
17669 : These bits are decoded and will assert UCLR ONLINE which is an input
17670 : to the ONLINE REG PAL. At the next P2 CLOCK L, ONLINE 0 L will be
17671 : negated.
17672 : The same sequence is repeated to clear online for each drive.
17673 : The DAR is cleared.
17674 : Then Drive 0 is again selected. When Drive 0 is selected at the ONLINE
17675 : REG PAL, output ONLINE H will reflect ONLINE 0 L. The IDC microcode
17676 : will then branch to an instruction that selects BEN1/DRIVE.ONLINE.
17677 : The outputs of the microsequencer will be:
17678 : BEN1 S2 H = H
17679 : BEN1 S1 H = H
17680 : BEN1 S0 H = L
17681 : These bits will output ONLINE H = L as NUA 1 H. The IDC microcode
17682 : will branch to an instruction that will send XFER REQ to flag the
17683 : CPU and will NOT increment the DAR. The CPU will then read and check
17684 : the DAR = 0.
17685 : Each drive is then checked to insure that each is cleared online.
17686 :
17687 : ERROR 2:
17688 : Drive 0 is selected by CSR <9:8> (see ERROR 1).
17689 : The IDC microcode will branch to a FUNC/SET.ONLINE instruction.
17690 : The outputs of the microsequencer will be:
17691 : FUNC 3 H = H
17692 : FUNC 2 H = L
17693 : FUNC 1 H = H
17694 : FUNC 0 H = L
17695 : These bits are decoded and will assert USET ONLINE L which is an
17696 : input to the ONLINE REG PAL. At the next P2 CLOCK L, ONLINE 0 L will
17697 : assert and approximately 20ns later ONLINE H will assert. The IDC
17698 : microcode will then branch to an instruction that selects
17699 : BEN1/DRIVE.ONLINE. The outputs of the microsequencer will be:
17700 : BEN1 S2 H = H
17701 : BEN1 S1 H = H
17702 : BEN1 S0 H = L
17703 : These bits will output ONLINE H = H as NUA 1 H. The IDC microcode
17704 : will branch to an instruction that will increment the DAR and then
17705 : send XFER REQ to flag the CPU. The CPU will then read and check
17706 : the DAR = 1.
17707 : ERROR 3:
17708 : Each of the other drives is then checked to insure that each remained
17709 : cleared online after ERROR 2.
17710 :
17711 : ERROR 4:
17712 : The DAR is cleared.
17713 : Drive 0 is selected (See ERROR 1). The IDC
17714 : microcode branches to a FUNC/CLR.ONLINE instruction. the outputs of the
17715 : microsequencer will be:
17716 : UFUNC 3 H = H
17717 : UFUNC 2 H = L
17718 : UFUNC 1 H = L
17719 : UFUNC 0 H = H
17720 : These bits are decoded and will assert UCLR ONLINE which is an input

```

[illegible]

ZZ-ENKCF-1.4 :  
: ENKCF.MCR  
: ENKCF.MIC

ENKCF.MIC

Test 1B ONLINE PAL15-MAR-83  
MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module  
Test 1B ONLINE PAL & BRANCH TEST, DR SEL CSR<9:8> (M8388 IDC MODULE)

Fiche 2 Frame L13

Sequence 373  
Rev 01.40 Page 367

ZZ-1  
:  
:

:17721 : to the ONLINE REG PAL. At the next P2 CLOCK L, ONLINE 0 L will be  
:17722 : negated. Approximately 20ns later ONLINE H will be negated.  
:17723 : The IDC microcode will then branch to an instruction that selects  
:17724 : BEN1/DRIVE.ONLINE. The outputs of the microsequencer will be:  
:17725 : BEN1 S2 H = H  
:17726 : BEN1 S1 H = H  
:17727 : BEN1 S0 H = L  
:17728 : These bits will output ONLINE H = L as NUA 1 H. The IDC microcode  
:17729 : will branch to an instruction that will send XFER REQ to flag the CPU  
:17730 : and will NOT increment the DAR. The CPU will then read and check  
:17731 : the DAR = 0.

:17732 :  
:17733 : The above sequence of ERRORS 2, 3, and 4 will be repeated with the  
:17734 : drive number that is selected in OTHER.DATA.  
:17735 :  
:17736 :--

:17737 :\*\*\*\*\*  
:17738 :T.1B:

|                 |        |                                 |                                |
|-----------------|--------|---------------------------------|--------------------------------|
| U OFF4, B65E,15 | :17739 | MOV LS[BEGIN.TEST] TO WR[0]     | ;SET BIT 15 IN WR[0] FOR       |
|                 | :17740 |                                 | ; CONTROL AND STATUS WORD      |
| U OFF5, 3E80,15 | :17741 | MOV WR[0] TO LS[CONTROL.STATUS] | ;SET BIT 15 IN CONTROL AND     |
|                 | :17742 |                                 | ; STATUS WORD - BIT 15         |
|                 | :17743 |                                 | ; INDICATES START OF TEST TO   |
|                 | :17744 |                                 | ; CONSOLE                      |
| U OFF6, 10E0,15 | :17745 | MISC [SET.CP.ATTN]              | ;ISSUE CPU ATTN TO CONSOLE     |
|                 | :17746 | WAIT.T1B.0:                     |                                |
| U OFF7, 08FF,74 | :17747 | JMP [WAIT.T1B.0]                | ;LOOP HERE WAITING FOR         |
|                 | :17748 |                                 | ; CONSOLE TO RESPOND           |
|                 | :17749 |                                 |                                |
| U OFF8, 65A0,15 | :17750 | CLR LS[PREVIOUS.ERROR]          | ;CLEAR PREVIOUS ERROR NUMBER   |
|                 | :17751 |                                 | ; IN LS                        |
|                 | :17752 |                                 |                                |
| U OFF9, B64A,15 | :17753 | MOV LS[IDC] TO WR[0]            | ;STORE MODULE CODE IN LS TO    |
| U OFFA, 3E8C,15 | :17754 | MOV WR[0] TO LS[MODULE.NUM]     | ; INDICATE IDC MODULE          |
|                 | :17755 |                                 |                                |
|                 | :17756 |                                 |                                |
| U OFFB, B640,15 | :17757 | MOV LS[#1] TO WR[0]             | ;SET WRO = 1                   |
| U OFFC, BE82,15 | :17758 | MOV WR[0] TO LS[ERROR.NUMBER]   | ;SET UP ERROR NUMBER = 1 IN LS |
|                 | :17759 |                                 |                                |
| U OFFD, 3776,15 | :17760 | MOV LS[DAR.MASK] TO WR[0]       | ;SET UP THE MASK FOR BITS      |
| U OFFE, 3E8A,15 | :17761 | MOV WR[0] TO LS[ERROR.MASK]     | ; <18:0>                       |
|                 | :17762 |                                 |                                |
| U OFFF, B670,15 | :17763 | MOV LS[OTHER.DATA] TO WR[0]     | ;SET UP SO OTHER DATA WILL     |
| U 1000, 3E80,15 | :17764 | MOV WR[0] TO LS[CONTROL.STATUS] | ; PRINT                        |
|                 | :17765 |                                 |                                |
| U 1001, 8872,EC | :17766 | JSR [DIAG.CODE]                 | ;FORCE THE IDC TO DIAG.IDLE    |
|                 | :17767 |                                 |                                |
|                 | :17768 |                                 |                                |
| U 1002, 2F85,15 | :17769 | CLR WR[2]                       | ;USED AS THE DRIVE NUM COUNTER |
|                 | :17770 | CLR.DRIVES.T1B:                 |                                |
| U 1003, 2004,15 | :17771 | MOV WR[2] TO WR[0]              | ;SHIFT THE DRIVE NUMBER INTO   |
| U 1004, 3646,95 | :17772 | MOV LS[#8] TO WR[1]             | ; BITS <9:8> TO WRITE TO THE   |
|                 | :17773 | SHIFT.T1B.1:                    | ; CSR                          |
| U 1005, 23D0,15 | :17774 | ASHL WR[0]                      |                                |
|                 | :17775 | DEC WR[1],                      |                                |

U 1  
U 1  
U 1  
U 1  
U 1  
U 1  
U 1  
U 1

|   |    |
|---|----|
| U | 10 |
| U | 10 |
| U | 10 |
| U | 10 |
| U | 10 |
|   |    |
| U | 10 |
|   |    |
| U | 10 |
| U | 10 |
| U | 10 |
|   |    |
| U | 10 |
| U | 10 |
| U | 10 |
| U | 10 |

```

:17831 :159:
:17832 :=*0*
:17833 :DIAG.WAIT1:
:17834 :
:17835 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:17836 : ; COMMAND FROM THE CPU
:17837 :-----
:17838 :15B:
:17839 :=*1*
:17840 :DIAG.WAIT2:
:17841 :
:17842 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:17843 : ; WAIT SOME MORE
:17844 :-----
:17845 :WAIT.IDC.T1B.1:
U 100D, DB00,1A :17846 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR IDC TO COMPLETE
U 100E, 8901,04 :17847 : JMP [IDC.DONE.T1B.1] ;IDC DONE
U 100F, 8900,D4 :17848 : JMP [WAIT.IDC.T1B.1] ;WAIT SOME MORE
:17849 :
:17850 :IDC.DONE.T1B.1:
U 1010, B700,15 :17851 : MOV LS[SETCSR.F0] TO WR[0] ;CLEAR THE FUNCTION BITS
U 1011, 17A0,15 :17852 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0]
U 1012, 90FA,15 :17853 : MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR XFER REQ
:17854 :
U 1013, 2045,15 :17855 : INC WR[2] ;INC THE DRIVE NUMBER
:17856 : CMP WR[2] WITH LS[#4], - ;LAST DRIVE BEEN CLEARED?
U 1014, CF45,35 :17857 : DT(LONG)&SET.ALU.CC
U 1015, 0900,31 :17858 : JMP.IF[NEQ] TO [CLR.DRIVES.T1B] ;JMP IF NOT DONE
:17859 :
:17860 :;ALL DRIVES <3:0> SHOULD NOW HAVE ONLINE CLEARED
:17861 :
U 1016, 2F85,15 :17862 : CLR WR[2] ;DRIVE NUM = 0
:17863 :
:17864 :TEST.DR.CLR.T1B:
U 1017, 2004,15 :17865 : MOV WR[2] TO WR[0] ;SHIFT THE DRIVE NUMBER INTO
U 1018, 3646,95 :17866 : MOV LS[#8] TO WR[1] ; BITS <9:8> TO WRITE TO THE
:17867 :SHIFT.T1B.2: ; CSR
U 1019, 23D0,15 :17868 : ASHL WR[0]
:17869 : DEC WR[1],
U 101A, A102,B5 :17870 : DT(LONG)&SET.ALU.CC
U 101B, 8901,91 :17871 : JMP.IF[NEQ] TO [SHIFT.T1B.2] ;WR[0] CONTAINS DR NUM IN<9:8>
U 101C, A001,95 :17872 : MOV WR[0] TO WR[3] ;SAVE THE INFO
:17873 :LOOP.T1B.1:
U 101D, 2F80,15 :17874 : CLR WR[0]
U 101E, 97A4,15 :17875 : PORT.WRITE PORT.ADRS[DAR] WITH WR[0] ;CLEAR THE DAR
:17876 :
U 101F, 3712,95 :17877 : MOV LS[SETCSR.CRDY] TO WR[1] ;SET UP THE CSR TO BRANCH TO
U 1020, B70A,15 :17878 : MOV LS[SETCSR.F5] TO WR[0] ; THE IDC ROUTINE THAT WILL
U 1021, AEC2,15 :17879 : BIS WR[1] TO WR[0] ; SELECT BEN1/ONLINE
U 1022, 2EC6,15 :17880 : BIS WR[3] TO WR[0] ;INCLUDE THE DRIVE NUMBER
U 1023, 17A0,15 :17881 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:17882 : MAINT=0,CRDY=1,F<2:0>=101
:17883 :;SELECT BEN1/DRIVE.ONLINE
:17884 :-----
:17885 :

```



```

17886 : *****
17887 : * IDC *
17888 : *****
17889 : -----
17890 : 018:
17891 : =000
17892 : DIAG.IDLE:
17893 :
17894 : BEN0/F0,
17895 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
17896 : BEN2/F2,
17897 : NAD/DIAG.IDLE
17898 : -----
17899 : 01D:
17900 : =101
17901 :
17902 : BEN0/CRDY.L,
17903 : BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
17904 : NAD/TEST.FUNC5 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
17905 : -----
17906 : 070:
17907 : =0*0
17908 : TEST.FUNC5:
17909 :
17910 : ONLINETST1.1:
17911 : BEN1/DRIVE.ONLINE, ;IS THE DRIVE SELECTED BY DRIVE
17912 : NAD/ONLINETST1.2 ; SELECT <1:0> ONLINE?
17913 : -----
17914 : 191:
17915 : =0*
17916 : ONLINETST1.2:
17917 :
17918 : NAD/XFER.REQ ;ONLINE NOT ASSERTED
17919 : -----
17920 : 193:
17921 : =1*
17922 : ONLINETST1.3:
17923 :
17924 : INCR.DAR/ON, ;ONLINE ASSERTED..INCR THE DAR
17925 : NAD/XFER.REQ ; AND SEND XFER REQ TO FLAG THE CPU
17926 : -----
17927 : 1D8:
17928 : XFER.REQ:
17929 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
17930 : NAD/DIAG.WAIT
17931 : -----
17932 : 1D7:
17933 : DIAG.WAIT:
17934 :
17935 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
17936 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
17937 : -----
17938 : 159:
17939 : =*0*
17940 : DIAG.WAIT1:

```

[illegible]

```

U 103E, B720,95 :17996 MOV LS[SETCSR.MAINT] TO WR[1] ;SET UP THE CSR TO BRANCH TO
U 103F, 370E,15 :17997 MOV LS[SETCSR.F7] TO WR[0] ; BRANCH TO THE IDC ROUTINE
U 1040, AEC2,15 :17998 BIS WR[1] TO WR[0] ; THAT WILL SET ONLINE
U 1041, 2EC6,15 :17999 BIS WR[3] TO WR[0]
U 1042, 17A0,15 :18000 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:18001 ;MAINT=1,CRDY=0,F<2:0>=111
:18002 ;USET ONLINE
:18003 -----
:18004 -----
:18005 :
:18006 : *****
:18007 : * IDC *
:18008 : *****
:18009 :
:18010 :018:
:18011 :=000
:18012 :DIAG.IDLE:
:18013 :
:18014 : BEN0/F0,
:18015 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:18016 : BEN2/F2,
:18017 : NAD/DIAG.IDLE
:18018 :
:18019 :01F:
:18020 :=111
:18021 :
:18022 : BEN0/CRDY.L,
:18023 : BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:18024 : NAD/TEST.FUNC7 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:18025 :
:18026 :07D:
:18027 :=1*1
:18028 :TEST.F7:
:18029 :
:18030 : FUNC/SET.ONLINE, ;SET ONLINE THE DRIVE SELECTED BY
:18031 : ; DRIVE SEL <1:0> (FOR ONLINE TESTS)
:18032 :
:18033 : LOAD.LOOP.CNTR.[OFF], ;LOAD LOOP CNTR = FF (FOR LP CNTR TST)
:18034 :
:18035 : NAD/XFER.REQ ;DO NOT INCR DAR BETWEEN HERE
:18036 : ; AND DIAG.IDLE
:18037 :
:18038 :1D8:
:18039 :XFER.REQ:
:18040 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:18041 : NAD/DIAG.WAIT
:18042 :
:18043 :1D7:
:18044 :DIAG.WAIT:
:18045 :
:18046 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:18047 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:18048 :
:18049 :159:
:18050 :=*0*
:18051 :DIAG.WAIT1:

```

```

18051 :
18052 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
18053 : ; COMMAND FROM THE CPU
18054 :-----
18055 :15B:
18056 :=*1*
18057 :DIAG.WAIT2:
18058 :
18059 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
18060 : ; WAIT SOME MORE
18061 :-----
18062 :WAIT.IDC.T1B.3:
U 1043, DB00,1A 18063 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR IDC TO COMPLETE
U 1044, 8904,64 18064 : JMP [IDC.DONE.T1B.3] ;IDC DONE
U 1045, 8904,34 18065 : JMP [WAIT.IDC.T1B.3] ;WAIT SOME MORE
18066 :
18067 :IDC.DONE.T1B.3:
U 1046, B700,15 18068 : MOV LS[SETCSR.F0] TO WR[0] ;CLEAR THE FUNCTION BITS
U 1047, 17A0,15 18069 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0]
U 1048, 90FA,15 18070 : MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR XFER REQ
18071 :
U 1049, B712,15 18072 : MOV LS[SETCSR.CRDY] TO WR[0] ;SET UP THE CSR TO BRANCH TO
U 104A, 370A,95 18073 : MOV LS[SETCSR.F5] TO WR[1] ; THE IDC ROUTINE THAT WILL
U 104B, AEC2,15 18074 : BIS WR[1] TO WR[0] ; SEL BEN1/ONLINE AND INCR DAR
18075 : ; IF IT IS ASSERTED
U 104C, 2EC6,15 18076 : BIS WR[3] TO WR[0] ;INCLUDE THE DRIVE NUMBER WR[3]
U 104D, 17A0,15 18077 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
18078 : ;MAINT=0,CRDY=1,F<2:0>=101
18079 :SELECT BEN1/DRIVE.ONLINE
18080 :-----
18081 :
18082 : *****
18083 : * IDC *
18084 : *****
18085 :-----
18086 :018:
18087 :=000
18088 :DIAG.IDLE:
18089 :
18090 : BEN0/F0,
18091 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
18092 : BEN2/F2,
18093 : NAD/DIAG.IDLE
18094 :-----
18095 :01D:
18096 :=101
18097 :
18098 : BEN0/CRDY.L,
18099 : BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
18100 : NAD/TEST.FUNC5 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
18101 :-----
18102 :070:
18103 :=0*0
18104 :TEST.FUNC5:
18105 :

```

```

18106 :ONLINETST1.1:
18107 : BEN1/DRIVE.ONLINE, ;IS THE DRIVE SELECTED BY DRIVE
18108 : NAD/ONLINETST1.2 ; SELECT <1:0> ONLINE?
18109 :-----
18110 :191:
18111 :=0*
18112 :ONLINETST1.2:
18113 :
18114 : NAD/XFER.REQ ;ONLINE NOT ASSERTED
18115 :-----
18116 :193:
18117 :=1*
18118 :ONLINETST1.3:
18119 :
18120 : INCR.DAR/ON, ;ONLINE ASSERTED..INCR THE DAR
18121 : NAD/XFER.REQ ; AND SEND XFER REQ TO FLAG THE CPU
18122 :-----
18123 :1D8:
18124 :XFER.REQ:
18125 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
18126 : NAD/DIAG.WAIT
18127 :-----
18128 :1D7:
18129 :DIAG.WAIT:
18130 :
18131 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
18132 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
18133 :-----
18134 :159:
18135 :=*0*
18136 :DIAG.WAIT1:
18137 :
18138 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
18139 : ; COMMAND FROM THE CPU
18140 :-----
18141 :15B:
18142 :=*1*
18143 :DIAG.WAIT2:
18144 :
18145 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
18146 : ; WAIT SOME MORE
18147 :-----
18148 :WAIT.IDC.T1B.4:
18149 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR IDC TO COMPLETE
18150 : JMP [IDC.DONE.T1B.4] ;IDC DONE
18151 : JMP [WAIT.IDC.T1B.4] ;WAIT SOME MORE
18152 :
18153 :IDC.DONE.T1B.4:
18154 : MOV LS[SETCSR.F0] TO WR[0] ;CLEAR THE FUNCTION BITS
18155 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0]
18156 : MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR XFER REQ
18157 :
18158 :READ.DAR.T1B.4:
18159 : MISC [SET.PORT.1.0] ;SELECT THE IDC
18160 : PORT.READ PORT.ADRS[DAR] ;SET UP TO READ THE DAR

```

U 104E, DB00,1A  
 U 104F, 8905,14  
 U 1050, 0904,E4

U 1051, B700,15  
 U 1052, 17A0,15  
 U 1053, 90FA,15

U 1054, 9090,15  
 U 1055, 1784,15

ZZ-  
 :  
 :

U 1

U 1

U 1

[illegible]

```

18216 :DIAG.IDLE:
18217 :
18218 : BEN0/F0,
18219 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
18220 : BEN2/F2,
18221 : NAD/DIAG.IDLE
18222 :-----
18223 :01D:
18224 :=101
18225 :
18226 : BEN0/CRDY.L,
18227 : BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
18228 : NAD/TEST.FUNC5 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
18229 :-----
18230 :070:
18231 :=0*0
18232 :TEST.FUNC5:
18233 :
18234 :ONLINETST1.1:
18235 : BEN1/DRIVE.ONLINE, ;IS THE DRIVE SELECTED BY DRIVE
18236 : NAD/ONLINETST1.2 ; SELECT <1:0> ONLINE?
18237 :-----
18238 :191:
18239 :=0*
18240 :ONLINETST1.2:
18241 :
18242 : NAD/XFER.REQ ;ONLINE NOT ASSERTED
18243 :-----
18244 :193:
18245 :=1*
18246 :ONLINETST1.3:
18247 :
18248 : INCR.DAR/ON, ;ONLINE ASSERTED..INCR THE DAR
18249 : NAD/XFER.REQ ; AND SEND XFER REQ TO FLAG THE CPU
18250 :-----
18251 :1D8:
18252 :XFER.REQ:
18253 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
18254 : NAD/DIAG.WAIT
18255 :-----
18256 :1D7:
18257 :DIAG.WAIT:
18258 :
18259 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
18260 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
18261 :-----
18262 :159:
18263 :=0*
18264 :DIAG.WAIT1:
18265 :
18266 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
18267 : ; COMMAND FROM THE CPU
18268 :-----
18269 :158:
18270 :=1*

```

```

:18271 :DIAG.WAIT2:
:18272 :
:18273 : NAD/DIAG.WAIT :XFER GRANT HAS NOT BEEN ISSUED....
:18274 : : WAIT SOME MORE
:18275 :-----
:18276 WAIT.IDC.T1B.5:
U 1076, DB00,1A :18277 SKIP.IF[NO.FAST.INTERRUPT] :WAIT FOR IDC TO COMPLETE
U 1077, 8907,94 :18278 JMP [IDC.DONE.T1B.5] :IDC DONE
U 1078, 8907,64 :18279 JMP [WAIT.IDC.T1B.5] :WAIT SOME MORE
:18280 :
:18281 IDC.DONE.T1B.5:
U 1079, B700,15 :18282 MOV LS[SETCSR.F0] TO WR[0] :CLEAR THE FUNCTION BITS
U 107A, 17A0,15 :18283 PORT.WRITE PORT.ADRS[CSR] WITH WR[0]
U 107B, 90FA,15 :18284 MISC2 [TRANSFER.GRANT] :SEND GRANT TO CLR XFER REQ
:18285 :
:18286 READ.DAR.T1B.5:
U 107C, 9090,15 :18287 MISC [SET.PORT.I.0] :SELECT THE IDC
U 107D, 1784,15 :18288 PORT.READ PORT.ADRS[DAR] :SET UP TO READ THE DAR
U 107E, 3B32,15 :18289 MOV PORT TO LS[PORT0] :READ THE DATA FROM DAR AND
:18290 : : SAVE IT IN LS
:18291 :
U 107F, B610,15 :18292 MOV LS[T8] TO WR[0] :GET THE DRIVE NUMBER UNDER
U 1080, BE88,15 :18293 MOV WR[0] TO LS[ADDRESS.DATA] : TEST FOR PRINTOUT
U 1081, 3732,15 :18294 MOV LS[PORT0] TO WR[0] :SET UP RECEIVED DATA
U 1082, 2F82,95 :18295 CLR WR[1] :SET UP THE EXPECTED DATA
U 1083, 0869,3C :18296 JSR [CHECK.RESULT] :CHECK THE DATA
U 1084, 890B,34 :18297 JMP [LOOP.T1B.3] :LOOP HERE IF ERR & LOE IS SET
:18298 :
:18299 CONTINUE.T1B:
U 1085, 3613,95 :18300 MOV LS[T9] TO WR[3] :RESTORE DRIVE NUMBER IN BITS
:18301 : : <9:8> TO WR[3]
U 1086, 0906,34 :18302 JMP [CHECK.NEXT.DR.T1B] :CHECK NEXT DRIVE
:18303 :
:18304 DONE.CHK.CLR.T1B:
:18305 :
U 1087, 3644,15 :18306 MOV LS[#4] TO WR[0]
U 1088, BE82,15 :18307 MOV WR[0] TO LS[ERROR.NUMBER] :SET UP FOR ERROR 4
:18308 :
:18309 :
U 1089, 2004,15 :18310 MOV WR[2] TO WR[0]
U 108A, 3646,95 :18311 MOV LS[#8] TO WR[1]
:18312 :
:18313 SHIFT.T1B.4:
U 108B, 23D0,15 :18313 ASHL WR[0]
:18314 : DEC WR[1],
U 108C, A102,85 :18315 DT(LONG)&SET.ALU.CC
U 108D, 0908,B1 :18316 JMP.IF[NEQ] TO [SHIFT.T1B.4] :WR[0] HAS DR NUM IN BITS <9:8>
:18317 :
U 108E, A001,95 :18318 MOV WR[0] TO WR[3] :SAV THE INFO
:18319 :
:18320 LOOP.T1B.4:
U 108F, 2F80,15 :18321 CLR WR[0]
U 1090, 97A4,15 :18322 PORT.WRITE PORT.ADRS[DAR] WITH WR[0] :CLEAR THE DAR
:18323 :
U 1091, 3720,15 :18324 MOV LS[SETCSR.MAINT] TO WR[0] :SET UP THE CSR TO BRANCH TO
U 1092, 370A,95 :18325 MOV LS[SETCSR.F5] TO WR[1] : THE IDC ROUTINE THAT WILL

```

ZZ-  
 :  
 :  
 U 1  
 U 1  
 U 1  
 U 1  
 U 1  
 U 1



```

U 1093, AEC2,15 :18326 BIS WR[1] TO WR[0] ; CLEAR ONLINE
U 1094, 2EC6,15 :18327 BIS WR[3] TO WR[0] ; INCLUDE THE DRIVE NUMBER
U 1095, 17A0,15 :18328 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ; WRITE THE CSR
:18329 ;MAINT=1,CRDY=0,F<2:0>=101
:18330 :UCLR ONLINE
:18331 -----
:18332 -----
:18333 :
:18334 : *****
:18335 : * IDC *
:18336 : *****
:18337 :-----
:18338 :018:
:18339 :=000
:18340 :DIAG.IDLE:
:18341 :
:18342 : BEN0/F0,
:18343 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:18344 : BEN2/F2,
:18345 : NAD/DIAG.IDLE
:18346 :-----
:18347 :01D:
:18348 :=101
:18349 :
:18350 : BEN0/CRDY.L,
:18351 : BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:18352 : NAD/TEST.FUNCS ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:18353 :-----
:18354 :075:
:18355 :=1*1
:18356 :TEST.F5:
:18357 : FUNC/CLR.ONLINE, ;CLR ONLINE FOR THE DRIVE SELECTED
:18358 : ; BY DRIVE SEL<1:0> (ONLINE PAL TESTS)
:18359 :
:18360 : UCMD/SET.FIFOA, ;SELECT FIFO A (USED FOR ANY TEST WHICH
:18361 : ; MUST USEL FIFO A)
:18362 : NAD/XFER.REQ ;DO NOT INCR DAR BETWEEN HERE
:18363 : ; AND DIAG.IDLE
:18364 :-----
:18365 :1D8:
:18366 :XFER.REQ:
:18367 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:18368 : NAD/DIAG.WAIT
:18369 :-----
:18370 :1D7:
:18371 :DIAG.WAIT:
:18372 :
:18373 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:18374 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:18375 :-----
:18376 :159:
:18377 :=*0*
:18378 :DIAG.WAIT1:
:18379 :
:18380 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND

```

```

U 1
U 1
U 1

U 1
U 1
U 1

U 1
U 1

```

```
:18381 : ; COMMAND FROM THE CPU
:18382 : -----
:18383 : 15B:
:18384 : =*1*
:18385 : DIAG.WAIT2:
:18386 :
:18387 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:18388 : ; WAIT SOME MORE
:18389 : -----
:18390 : WAIT.IDC.T1B.6:
U 1096, DB00,1A :18391 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR IDC TO COMPLETE
U 1097, 0909,94 :18392 : JMP [IDC.DONE.T1B.6] ;IDC DONE
U 1098, 0909,64 :18393 : JMP [WAIT.IDC.T1B.6] ;WAIT SOME MORE
:18394 :
:18395 : IDC.DONE.T1B.6:
U 1099, B700,15 :18396 : MOV LS[SETCSR.F0] TO WR[0] ;CLEAR THE FUNCTION BITS
U 109A, 17A0,15 :18397 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0]
U 109B, 90FA,15 :18398 : MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR XFER REQ
:18399 :
:18400 :
U 109C, 3712,95 :18401 : MOV LS[SETCSR.CRDY] TO WR[1] ;SET UP THE CSR TO BRANCH TO
U 109D, B70A,15 :18402 : MOV LS[SETCSR.F5] TO WR[0] ; THE IDC ROUTINE THAT WILL
U 109E, AEC2,15 :18403 : BIS WR[1] TO WR[0] ; SEL BEN1/ONLINE AND INCR DAR
:18404 : ; IF ASSERTED
U 109F, 2EC6,15 :18405 : BIS WR[3] TO WR[0] ;INCLUDE THE DRIVE NUMBER
U 10A0, 17A0,15 :18406 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:18407 : ;MAINT=0,CRDY=1,F<2:0>=101
:18408 : ;SELECT BEN1/DRIVE.ONLINE
:18409 : -----
:18410 : -----
:18411 : *****
:18412 : * IDC *
:18413 : *****
:18414 : -----
:18415 : 018:
:18416 : =000
:18417 : DIAG.IDLE:
:18418 :
:18419 : BEN0/F0,
:18420 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:18421 : BEN2/F2,
:18422 : NAD/DIAG.IDLE
:18423 : -----
:18424 : 01D:
:18425 : =101
:18426 :
:18427 : BEN0/CRDY.L,
:18428 : BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:18429 : NAD/TEST.FUNC5 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:18430 : -----
:18431 : 070:
:18432 : =0*0
:18433 : TEST.FUNC5:
:18434 :
:18435 : ONLINETST1.1:
```

ZZ-  
:  
:  
U 1  
U 1  
U 1  
U 1  
U 1  
U 1  
U 1  
U 1  
U 1  
U 1

[illegible]

[illegible]

|   |   |
|---|---|
| U | 1 |
| U | 1 |
| U | 1 |
| U | 1 |

```
:18601 :DIAG.IDLE:
:18602 :
:18603 : BEN0/F0,
:18604 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:18605 : BEN2/F2,
:18606 : NAD/DIAG.IDLE
:18607 :-----
:18608 :01D:
:18609 :=101
:18610 :
:18611 : BEN0/CRDY.L,
:18612 : BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:18613 : NAD/TEST.FUNC5 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:18614 :-----
:18615 :070:
:18616 :=0*0
:18617 :TEST.FUNC5:
:18618 :
:18619 :ONLINETST1.1:
:18620 : BEN1/DRIVE.ONLINE, ;IS THE DRIVE SELECTED BY DRIVE
:18621 : NAD/ONLINETST1.2 ; SELECT <1:0> ONLINE?
:18622 :-----
:18623 :191:
:18624 :=0*
:18625 :ONLINETST1.2:
:18626 :
:18627 : NAD/XFER.REQ ;ONLINE NOT ASSERTED
:18628 :-----
:18629 :193:
:18630 :=1*
:18631 :ONLINETST1.3:
:18632 :
:18633 : INCR.DAR/ON, ;ONLINE ASSERTED..INCR THE DAR
:18634 : NAD/XFER.REQ ; AND SEND XFER REQ TO FLAG THE CPU
:18635 :-----
:18636 :1D8:
:18637 :XFER.REQ:
:18638 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:18639 : NAD/DIAG.WAIT
:18640 :-----
:18641 :1D7:
:18642 :DIAG.WAIT:
:18643 :
:18644 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:18645 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:18646 :-----
:18647 :159:
:18648 :=*0*
:18649 :DIAG.WAIT1:
:18650 :
:18651 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:18652 : ; COMMAND FROM THE CPU
:18653 :-----
:18654 :15B:
:18655 :=*1*
```

U  
U  
U  
U  
U  
U

```

:18656 ;DIAG.WAIT2:
:18657
:18658 NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:18659 ; WAIT SOME MORE
:18660 -----
:18661 WAIT.IDC.T1B.9:
U 10C3, DB00,1A :18662 SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR IDC TO COMPLETE
U 10C4, 090C,64 :18663 JMP [IDC.DONE.T1B.9] ;IDC DONE
U 10C5, 090C,34 :18664 JMP [WAIT.IDC.T1B.9] ;WAIT SOME MORE
:18665
:18666 IDC.DONE.T1B.9:
U 10C6, B700,15 :18667 MOV LS[SETCSR.F0] TO WR[0] ;CLEAR THE FUNCTION BITS
U 10C7, 17A0,15 :18668 PORT.WRITE PORT.ADRS[CSR] WITH WR[0]
U 10C8, 90FA,15 :18669 MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR XFER REQ
:18670
:18671 READ.DAR.T1B.9:
U 10C9, 9090,15 :18672 MISC [SET.PORT.1.0] ;SELECT THE IDC
U 10CA, 1784,15 :18673 PORT.READ PORT.ADRS[DAR] ;SET UP TO READ THE DAR
U 10CB, 3B32,15 :18674 MOV PORT TO LS[PORT0] ;READ THE DATA FROM DAR AND
:18675 ; SAVE IT IN LS
:18676
U 10CC, B610,15 :18677 MOV LS[T8] TO WR[0] ;GET THE DRIVE NUMBER UNDER
U 10CD, BE88,15 :18678 MOV WR[0] TO LS[ADDRESS.DATA] ; TEST FOR PRINTOUT
U 10CE, 3732,15 :18679 MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
U 10CF, 2F82,95 :18680 CLR WR[1] ;SET UP THE EXPECTED DATA
U 10D0, 0869,3C :18681 JSR [CHECK.RESULT] ;CHECK THE DATA
U 10D1, 890B,34 :18682 JMP [LOOP.T1B.3] ;LOOP HERE IF ERR & LOE IS SET
U 10D2, 8908,54 :18683 JMP [CONTINUE.T1B] ;CONTINUE TEST
:18684
:18685 CLEANUP.T1B:
U 10D3, 8875,3C :18686 JSR [CLEANUP] ;CLEANUP CONDITIONS SET BY THIS
:18687 ; TEST
:18688
:18689 END.T1B: ;END TEST
:18690
:18691
:18692

```

```

:18693 .PAGE 'Test 1C ONLINE PAL & BRANCH TEST WITH USEL.DRV (M8388 IDC MODULE)''
:18694 ++
:18695
:18696 .Test Description:
:18697
:18698 This test is designed to verify the ONLINE REGISTER PAL and a
:18699 portion of the drive select logic.
:18700
:18701 First all of the drives will be cleared online using CSR <9:8>,
:18702 (DS1, DS0) as the drive select.
:18703
:18704 Then Drive 0 will be selected by USEL DRV <1:0> and it will
:18705 be set online. Drive 0 will then be selected by CSR <9:8>
:18706 and checked that it is online.
:18707 This sequence will be repeated for each drive <3:0>.
:18708
:18709 .Assumptions:
:18710
:18711 It is assumed that the ONLINE TEST with CSR <DS1> <DS0> has run
:18712 successfully.
:18713
:18714 .Test Steps:
:18715
:18716 1. Set up error mask, error number and module number in LS
:18717 (for error printouts) and clear the previous error number in LS.
:18718 2. Force the IDC microcode to DIAG.IDLE.
:18719 3. Set CSR <DS1> <DS0> = 0.
:18720 4. Set up the CSR to branch to the correct microcode routine.
:18721
:18722 *****
:18723 * IDC *
:18724 *****
:18725
:18726 IDC1. Set FUNC BITS <2:0> = 9, CLR.ONLINE.
:18727 IDC2. Issue XFER REQ to flag the CPU and go to DIAG.IDLE.
:18728
:18729 5. Wait for a XFER REQ from IDC.
:18730 6. Clear CSR F<2:0> and send XFER GRANT to clear the XFER REQ.
:18731 7. Move the next drive number to CSR <DS1> <DS0> and perform
:18732 Steps 4 - 7 until all drives have been cleared online.
:18733 8. Set WR[2] = 0 to indicate the drive number under test.
:18734 9. Set up the CSR to branch to the correct microcode routine.
:18735
:18736 *****
:18737 * IDC *
:18738 *****
:18739
:18740 IDC3. Load the Loop Counter with 0.
:18741
:18742 10. Wait for a XFER REQ from IDC.
:18743 11. Clear CSR F<2:0> and send XFER GRANT to clear the XFER REQ.
:18744 12. Set up the CSR to branch to the correct microcode routine.
:18745
:18746 *****
:18747

```



```

:18748 : * IDC *
:18749 : *
:18750 : IDC4. Assert USEL.DRV.
:18751 : IDC5. Set FUNL BITS <35:32> = A, SET.ONLINE.
:18752 : IDC6. Issue XFER REQ and go to DIAG.IDLE.
:18753 :
:18754 : 13. Wait for a XFER REQ from IDC.
:18755 : 14. Clear CSR F<2:0> and send XFER GRANT to clear the XFER REQ.
:18756 : 15. Use CSR <DS1> <DS0> to select the drive number.
:18757 : 16. Clear the DAR.
:18758 : 17. Set up the CSR to branch to the correct microcode routine.
:18759 :
:18760 :
:18761 : *
:18762 : * IDC *
:18763 : *
:18764 :
:18765 : IDC7. Select BEN1/ONLINE H.
:18766 : IDC8. If ONLINE H is asserted then increment the DAR.
:18767 : IDC9. Issue XFER REQ to flag the CPU and go to DIAG.IDLE.
:18768 :
:18769 : 18. Wait for a XFER REQ from IDC.
:18770 : 19. Clear CSR F<2:0> and send XFER GRANT to clear the XFER REQ.
:18771 : 20. Check the DAR = 1.
:18772 : 21. Set up the CSR to branch to the correct microcode routine.
:18773 :
:18774 : *
:18775 : * IDC *
:18776 : *
:18777 :
:18778 : IDC10. Increment the Loop Counter to select the next drive.
:18779 :
:18780 : 22. Wait for a XFER REQ from IDC.
:18781 : 23. Clear CSR F<2:0> and send XFER GRANT to clear the XFER REQ.
:18782 : 24. Increment the counter WR[2].
:18783 : 25. Repeat Steps 12 - 25 until all drives have been tested.
:18784 :
:18785 : Errors:
:18786 :
:18787 : Printout:
:18788 : EXPECTED RECEIVED OTHER DATA
:18789 : DAR DAR WR[2]
:18790 : Drive Number
:18791 :
:18792 : ERROR 1 - The drive number in WR[2] was not online after USET.ONLINE
:18793 : was asserted with this drive selected by USEL <1:0>.
:18794 :
:18795 : Debug:
:18796 :
:18797 : ERROR 1:
:18798 : All of the drives will be cleared online using CSR<9:8> as the drive
:18799 : select.
:18800 : The IDC microcode will then branch to a LOAD.LOOP.CNTR.[00]
:18801 : instruction. The outputs of the microsequencer will be:
:18802 : ULOAD LPCNTR L = L

```

U U U U U U U U

```

:18858 WAIT.T1C.0:
U 10D7, 090D,74 :18859 JMP [WAIT.T1C.0] ;LOOP HERE WAITING FOR
:18860 ; CONSOLE TO RESPOND
:18861
U 10D8, 65A0,15 :18862 CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
:18863 ; IN LS
:18864
U 10D9, B64A,15 :18865 MOV LS[IDC] TO WR[0] ;STORE MODULE CODE IN LS TO
U 10DA, 3E8C,15 :18866 MOV WR[0] TO LS[MODULE.NUM] ; INDICATE IDC MODULE
:18867
:18868
U 10DB, B640,15 :18869 MOV LS[#1] TO WR[0] ;SET WRO = 1
U 10DC, BE82,15 :18870 MOV WR[0] TO LS[ERROR.NUMBER] ;SET UP ERROR NUMBER = 1 IN LS
:18871
:18872
U 10DD, 3776,15 :18872 MOV LS[DAR.MASK] TO WR[0]
U 10DE, 3E8A,15 :18873 MOV WR[0] TO LS[ERROR.MASK] ;SET UP ERROR MASK FOR THE DAR
:18874
:18875
U 10DF, B670,15 :18875 MOV LS[OTHER.DATA] TO WR[0] ;SET UP THE CONTROL.STATUS
U 10E0, 3E80,15 :18876 MOV WR[0] TO LS[CONTROL.STATUS] ; TO PRINT OTHER DATA ON ERROR
:18877
:18878
U 10E1, 8872,EC :18878 JSR [DIAG.CODE]
:18879
:18880 ;CLEAR ONLINE FOR ALL DRIVES <3:0>
:18881
:18882
U 10E2, 2F85,15 :18882 CLR WR[2] ;USED AS THE DRIVE NUM COUNTER
:18883 CLR.DRIVES.T1C:
:18884 MOV WR[2] TO WR[0] ;SHIFT THE DRIVE NUMBER INTO
U 10E3, 2004,15 :18884 MOV LS[#8] TO WR[1] ; BITS <9:8> TO WRITE TO THE
U 10E4, 3646,95 :18885 SHIFT.T1C.1: ; CSR
:18886
:18887
U 10E5, 23D0,15 :18887 ASHL WR[0]
:18888 DEC WR[1],
:18889 DT(LONG)&SET.ALU.CC
U 10E6, A102,B5 :18889 JMP.IF[NEQ] TO [SHIFT.T1C.1] ;WR[0] CONTAINS DR NUM IN<9:8>
U 10E7, 890E,51 :18890
:18891
:18892
U 10E8, B720,95 :18892 MOV LS[SETCSR.MAINT] TO WR[1] ;SET UP THE CSR TO BRANCH TO
U 10E9, AEC2,15 :18893 BIS WR[1] TO WR[0] ; THE IDC ROUTINE THAT WILL
U 10EA, 370A,95 :18894 MOV LS[SETCSR.F5] TO WR[1] ; CLR ONLINE AND THEN SEND
U 10EB, AEC2,15 :18895 BIS WR[1] TO WR[0] ; XFER REQ
U 10EC, 17A0,15 :18896 PORT.WRITE PORT.ADRS[CSR] WITH-WR[0] ;WRITE THE CSR
:18897 ;MAINT=1,CRDY=0,F<2:0>=101
:18898 ;UCLR ONLINE
:18899 -----
:18900 -----
:18901 *****
:18902 * IDC *
:18903 *****
:18904 -----
:18905 :018:
:18906 :=000
:18907 :DIAG.IDLE:
:18908
:18909 :BEN0/F0,
:18910 :BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:18911 :BEN2/F2,
:18912 :NAD/DIAG.IDLE

```

[illegible]

```

U 10F3, 2045,15 :18968 INC WR[2] ;INC THE DRIVE NUMBER
 :18969 CMP WR[2] WITH LS[4], ;LAST DRIVE BEEN CLEARED?
U 10F4, CF45,35 :18970 DT(LONG)&SET.ALU.CC
U 10F5, 890E,31 :18971 JMP.IF[NEQ] TO [CLR.DRIVES.T1C] ;JMP IF NOT DONE
 :18972
 :18973 ;ALL DRIVES <3:0> SHOULD NOW HAVE ONLINE CLEARED
 :18974
U 10F6, 2F85,15 :18975 CLR WR[2] ;CLEAR THE DRIVE NUMBER
U 10F7, B720,95 :18976 MOV LS[SETCSR.MAINT] TO WR[1] ;SET UP THE CSR TO BRANCH TO
U 10F8, B70C,15 :18977 MOV LS[SETCSR.F6] TO WR[0] ; THE IDC ROUTINE THAT WILL
U 10F9, AEC2,15 :18978 BIS WR[1] TO WR[0] ; LOAD LPCNTR = 0(DR NUM =0)
U 10FA, 17A0,15 :18979 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
 :18980 ;MAINT=1,CRDY=0,F<2:0>=110
 :18981 ;LOAD THE LOOP CNTR = 00
 :18982 -----
 :18983 -----
 :18984 *****
 :18985 * IDC *
 :18986 *****
 :18987 -----
 :18988 :018:
 :18989 :=000
 :18990 :DIAG.IDLE:
 :18991
 :18992 BEN0/F0,
 :18993 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
 :18994 BEN2/F2,
 :18995 NAD/DIAG.IDLE
 :18996 -----
 :18997 :01E:
 :18998 :=110
 :18999
 :19000 BEN0/CRDY.L,
 :19001 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
 :19002 NAD/TEST.FUNC6 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
 :19003 -----
 :19004 :077:
 :19005 :=1*1
 :19006 :TEST.F6:
 :19007 ;DO NOT INCR DAR BETWEEN HERE AND
 :19008 ; DIAG.IDLE OR DIAG.WAIT
 :19009
 :19010 LOAD.LOOP.CNTR.[00], ;LOAD LOOP CNTR WITH 00
 :19011
 :19012 UCMD/SET.DLT, ;SET DLT FOR DLT TEST
 :19013
 :19014 FUNC/SET.INT, ;SET INT FOR INTERRUPT TEST
 :19015
 :19016 NAD/XFER.REQ ;SEND XFER REQ TO FLAG THE CPU
 :19017 -----
 :19018 :1D8:
 :19019 :XFER.REQ:
 :19020 UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
 :19021 NAD/DIAG.WAIT
 :19022 -----

```

[illegible]

```

19078 : BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
19079 : NAD/TEST.FUNC5 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
19080 : -----
19081 : 071:
19082 : =0*1
19083 : ONLINETST2.1:
19084 :
19085 : SEL.DRV/UPROG, ;USE THE LOOP COUNTER TO SELECT THE
19086 : FUNC/SET.ONLINE, ; DRIVE NUMBER AND SET IT ONLINE
19087 : NAD/XFER.REQ ;SEND XFER REQ TO FLAG THE CPU
19088 : -----
19089 : 1D8:
19090 : XFER.REQ:
19091 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
19092 : NAD/DIAG.WAIT
19093 : -----
19094 : 1D7:
19095 : DIAG.WAIT:
19096 :
19097 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
19098 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
19099 : -----
19100 : 159:
19101 : =*0*
19102 : DIAG.WAIT1:
19103 :
19104 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
19105 : ; COMMAND FROM THE CPU
19106 : -----
19107 : 15B:
19108 : =*1*
19109 : DIAG.WAIT2:
19110 :
19111 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
19112 : ; WAIT SOME MORE
19113 : -----
19114 : WAIT.IDC.T1C.3:
19115 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR IDC TO COMPLETE
19116 : JMP [IDC.DONE.T1C.3] ;IDC DONE
19117 : JMP [WAIT.IDC.T1C.3] ;WAIT SOME MORE
19118 :
19119 : IDC.DONE.T1C.3:
19120 : MOV LS[SETCSR.F0] TO WR[0] ;CLEAR THE FUNCTION BITS
19121 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0]
19122 : MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR XFER REQ
19123 :
19124 : ;PUT THE DRIVE NUMBER IN CSR <9:8>
19125 :
19126 : MOV WR[2] TO WR[0] ;GET THE DRIVE NUMBER
19127 : MOV LS[#8] TO WR[1]
19128 : SHIFT.T1C.2:
19129 : ASHL WR[0] ;SHIFT IT INTO BITS <9:8>
19130 : DEC WR[1]
19131 : DT(LONG)&SET.ALU.CC
19132 : JMP.IF[NEQ] TO [SHIFT.T1C.2]

```

U 1103, DB00,1A  
 U 1104, 8910,64  
 U 1105, 8910,34

U 1106, B700,15  
 U 1107, 17A0,15  
 U 1108, 90FA,15

U 1109, 2004,15  
 U 110A, 3646,95

U 110B, 23D0,15

U 110C, A102,B5  
 U 110D, 0910,B1

```

;19133 ;WR[0] NOW HAS DR NUM IN <9:8>
U 110E, A001,95 :19134 MOV WR[0] TO WR[3] ;SAVE IT IN WR[3]
:19135
:19136 ;NOW CHECK IF DRIVE IS ONLINE USING THE CSR TO SELECT THE DRIVE
:19137
U 110F, 2F80,15 :19138 CLR WR[0]
U 1110, 97A4,15 :19139 PORT.WRITE PORT.ADRS[DAR] WITH WR[0] ;CLR THE DAR
:19140
U 1111, 3712,95 :19141 MOV LS[SETCSR.CRDY] TO WR[1] ;SET UP THE CSR TO BRANCH TO
U 1112, B70A,15 :19142 MOV LS[SETCSR.F5] TO WR[0] ; THE IDC ROUTINE THAT WILL
U 1113, AEC2,15 :19143 BIS WR[1] TO WR[0] ; SEL BEN1/ONLINE H
U 1114, 2EC6,15 :19144 BIS WR[3] TO WR[0] ;INCLUDE THE DRIVE NUMBER
U 1115, 17A0,15 :19145 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:19146 ;MAINT=0,CRDY=1,F<2:0>=101
:19147 ;SELECT BEN1/DRIVE.ONLINE
:19148 -----
:19149 -----
:19150 *****
:19151 * IDC *
:19152 *****
:19153 -----
:19154 :018:
:19155 =000
:19156 DIAG.IDLE:
:19157
:19158 BEN0/F0,
:19159 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:19160 BEN2/F2,
:19161 NAD/DIAG.IDLE
:19162 -----
:19163 :01D:
:19164 =101
:19165
:19166 BEN0/CRDY.L,
:19167 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:19168 NAD/TEST.FUNC5 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:19169 -----
:19170 :070:
:19171 =0*0
:19172 TEST.FUNC5:
:19173
:19174 ONLINETST1.1:
:19175 BEN1/DRIVE.ONLINE, ;IS THE DRIVE SELECTED BY DRIVE
:19176 NAD/ONLINETST1.2 ; SELECT <1:0> ONLINE?
:19177 -----
:19178 :191:
:19179 =0*
:19180 ONLINETST1.2:
:19181
:19182 NAD/XFER.REQ ;ONLINE NOT ASSERTED
:19183 -----
:19184 :193:
:19185 =1*
:19186 ONLINETST1.3:
:19187

```



```

:19188 : INCR.DAR/ON, ;ONLINE ASSERTED..INCR THE DAR
:19189 : NAD/XFER.REQ ; AND SEND XFER REQ TO FLAG THE CPU
:19190 : -----
:19191 : 1D8:
:19192 : XFER.REQ:
:19193 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:19194 : NAD/DIAG.WAIT
:19195 : -----
:19196 : 1D7:
:19197 : DIAG.WAIT:
:19198 :
:19199 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:19200 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:19201 : -----
:19202 : 159:
:19203 : =*0*
:19204 : DIAG.WAIT1:
:19205 :
:19206 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:19207 : ; COMMAND FROM THE CPU
:19208 : -----
:19209 : 15B:
:19210 : =*1*
:19211 : DIAG.WAIT2:
:19212 :
:19213 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:19214 : ; WAIT SOME MORE
:19215 : -----
:19216 : WAIT.IDC.T1C.4:
U 1116, DB00,1A :19217 : SKIP.[IF[NO.FAST.INTERRUPT] ;WAIT FOR IDC TO COMPLETE
U 1117, 0911,94 :19218 : JMP [IDC.DONE.T1C.4] ;IDC DONE
U 1118, 0911,64 :19219 : JMP [WAIT.IDC.T1C.4] ;WAIT SOME MORE
:19220 :
:19221 : IDC.DONE.T1C.4:
U 1119, B700,15 :19222 : MOV LS[SETCSR.F0] TO WR[0] ;CLEAR THE FUNCTION BITS
U 111A, 17A0,15 :19223 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0]
U 111B, 90FA,15 :19224 : MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR XFER REQ
:19225 :
:19226 : READ.DAR.T1C.1:
U 111C, 9090,15 :19227 : MISC [SET.PORT.1.0] ;SELECT THE IDC
U 111D, 1784,15 :19228 : PORT.READ PORT.ADRS[DAR] ;SET UP TO READ THE DAR
U 111E, 3B32,15 :19229 : MOV PORT TO LS[PORT0] ;READ THE DATA FROM DAR AND
:19230 : ; SAVE IT IN LS
U 111F, 3E89,15 :19231 : MOV WR[2] TO LS[ADDRESS.DATA] ;SET UP THE DR NUM FOR PRINTOUT
U 1120, 3732,15 :19232 : MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
U 1121, 3640,95 :19233 : MOV LS[#1] TO WR[1] ;SET UP THE EXPECTED DATA
U 1122, 0869,3C :19234 : JSR [CHECK.RESULT] ;CHECK THE DATA
U 1123, 0910,14 :19235 : JMP [LOOP.T1C.1] ;LOOP HERE IF ERR & LOE IS SET
:19236 :
U 1124, 2045,15 :19237 : INC WR[2] ;NEXT DRIVE NUMBER
:19238 : CMP LS[#4] WITH WR[2], ;ALL DRIVES BEEN TESTED?
U 1125, 4E45,35 :19239 : DT(LONG)&SET.ALU.CC
U 1126, 8913,29 :19240 : JMP.[IF[EQL] TO [CLEANUP.T1C] ;JMP IF DONE
:19241 :
:19242 : ;INCR THE LOOP COUNTER TO INCR THE USEL DR LINES

```

B  
C  
D  
E  
F  
G  
H  
I  
J  
K  
L  
M  
N  
O  
P  
Q  
R  
S  
T  
U  
V  
W  
X  
Y  
Z



```

:19298 :XFER.REQ:
:19299 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:19300 : NAD/DIAG.WAIT
:19301 :-----
:19302 :1D7:
:19303 :DIAG.WAIT:
:19304 :
:19305 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:19306 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:19307 :-----
:19308 :159:
:19309 :=*0*
:19310 :DIAG.WAIT1:
:19311 :
:19312 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:19313 : ; COMMAND FROM THE CPU
:19314 :-----
:19315 :15B:
:19316 :=*1*
:19317 :DIAG.WAIT2:
:19318 :
:19319 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:19320 : ; WAIT SOME MORE
:19321 :-----
:19322 :WAIT.IDC.T1C.5:
:19323 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR IDC TO COMPLETE
:19324 : JMP [IDC.DONE.T1C.5] ;IDC DONE
:19325 : JMP [WAIT.IDC.T1C.5] ;WAIT SOME MORE
:19326 :
:19327 :IDC.DONE.T1C.5:
:19328 : MOV LS[SETCSR.F0] TO WR[0] ;CLEAR THE FUNCTION BITS
:19329 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0]
:19330 : MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR XFER REQ
:19331 :
:19332 : JMP [TEST.NEXT.DR.T1C] ;GO CHECK THE NEXT DRIVE
:19333 :
:19334 :CLEANUP.T1C:
:19335 : JSR [CLEANUP] ;CLEANUP THE CONDITIONS SET
:19336 : ; BY THIS TEST
:19337 :END.T1C: ;END TEST
:19338 :
:19339 :

```

U 112B, DB00,1A :19323  
 U 112C, 8912,E4 :19324  
 U 112D, 8912,B4 :19325  
 :19326  
 U 112E, B700,15 :19327  
 U 112F, 17A0,15 :19328  
 U 1130, 90FA,15 :19329  
 :19330  
 U 1131, 0910,14 :19331  
 :19332  
 :19333  
 U 1132, 8875,3C :19334  
 :19335  
 :19336  
 :19337  
 :19338  
 :19339

:19340  
:19341  
:19342  
:19343  
:19344  
:19345  
:19346  
:19347  
:19348  
:19349  
:19350  
:19351  
:19352  
:19353  
:19354  
:19355  
:19356  
:19357  
:19358  
:19359  
:19360  
:19361  
:19362  
:19363  
:19364  
:19365  
:19366  
:19367  
:19368  
:19369  
:19370  
:19371  
:19372  
:19373  
:19374  
:19375  
:19376  
:19377  
:19378  
:19379  
:19380  
:19381  
:19382  
:19383  
:19384  
:19385  
:19386  
:19387  
:19388  
:19389  
:19390  
:19391  
:19392  
:19393  
:19394

PAGE "Test 1D DATA BUS IN AND DATA BUS OUT TEST (M8388 IDC MODULE)"  
++  
Test Description:  
  
This test will verify the integrity of data bus in and data bus out.  
A portion of the I/O BUS IN LATCHES, I/O BUS OUT LATCHES and location 0  
of FIFO A and FIFO B will also be verified in this test.  
Only one location in FIFO A and FIFO B will be used to enable better  
isolation.  
The CPU will do a WRITE.DATA.BYTE of the data pattern to FIFO A  
address 0.  
The CPU will then read FIFO A address 0 and then compare the write and  
read data to insure that the write and the read functioned correctly.  
The data patterns that will be used are:  
  
Pattern 1 - AA  
Pattern 2 - 55  
Pattern 3 - 33  
Pattern 4 - 0F  
FIFO B will be selected after FIFO A has been verified using all of  
the data patterns.  
  
Assumptions:  
  
It is assumed that the CPU and the Y Bus are fully operational.  
  
Test Steps:  
  
1. Set up error mask, error number and module number in LS  
(for error printouts) and clear the previous error number in LS.  
2. Select FIFO A.  
3. Get the data pattern from LS.  
4. Move the data pattern to WR 0.  
5. Clear the FIFO Address Counter.  
6. Execute a WRITE.DATA.BYTE to the FIFO.  
7. Clear the FIFO Address Counter.  
8. Execute a READ.BYTE from the FIFO.  
9. Check the result.  
10. Get the next data pattern from LS and go to Step 3 until all data  
patterns have been tested using FIFO A.  
11. Get the next data pattern and Select FIFO B.  
12. Repeat Steps 3 - 9 until all data patterns have been tested  
using FIFO B.  
  
Errors:  
  
ERROR 1 - The byte data path failed using FIFO A.  
  
Printout:  
  

EXPECTED

RECEIVED

FIFO A
FIFO A

  
ERROR 2 - The byte data path failed using FIFO B.

```

:19395 :
:19396 :
:19397 :
:19398 :
:19399 :
:19400 :
:19401 :
:19402 :
:19403 :
:19404 :
:19405 :
:19406 :
:19407 :
:19408 :
:19409 :
:19410 :
:19411 :
:19412 :
:19413 :
:19414 :
:19415 :
:19416 :
:19417 :
:19418 :
:19419 :
:19420 :
:19421 :
:19422 :
:19423 :
:19424 :
:19425 :
:19426 :
:19427 :
:19428 :
:19429 :
:19430 :
:19431 :
:19432 :
:19433 :
:19434 :
:19435 :
:19436 :
:19437 :
:19438 :
:19439 :
:19440 :
:19441 :
:19442 :
:19443 :
:19444 :
:19445 :
:19446 :
:19447 :
:19448 :
:19449 :

```

Printout:

| EXPECTED | RECEIVED |
|----------|----------|
| FIFO B   | FIFO B   |

Debug:

ERROR 1:

SEL.FIFO.A:

FIFO A is selected by the port by a PORT.CONTROL [SEL.FIFO.A] instruction. This instruction is decoded by the PORT INTERFACE REG PAL. The inputs will be:

- CSR 17 H = H
- CSR 14 H = H
- CSR 12 H = H
- CSR 11 H = H
- CSR 10 H = L
- PORT INSTR H = H
- CPU P2 H = H

When the PAL is clocked by CPU CLOCK H, PSEL FIFO A H will assert.

WRITE.DATA.BYTE:

This instruction will be decoded in the PORT RD/WR DECODE PAL. The inputs to this PAL will be:

- CSR 17 H = H
- CSR 14 H = L
- CSR 13 H = H
- CSR 12 H = L
- CSR 11 H = H
- CSR 10 H = L
- PORT INSTR H = H
- CPU P2 H = H

This combination of inputs will assert BYTE L and WRITE DATA L, and STROBE DATA H. BYTE L and WRITE DATA L are inputs to the PORT USEQ A & B PALS. This combination of inputs will assert PWRITE FIFO H and PENB INREG B0 L. PENB INREG B0 L and STROBE DATA H will enable the data from the I/O Bus onto BUS IN. PWRITE FIFO H will be ANDED with PSEL FIFOA H and PORT CLK L. This combination of signals will assert WRITE FIFOA L which will write the data which is on BUS IN into FIFOA.

The CPU will then select FIFO A and issue a CLEAR.FIFO.ADRS which will clear the address lines of FIFOA.

READ.DATA.BYTE:

To read a byte of data from the FIFO, the CPU must first issue a MISC [SET.PORT.I.O] which will assert SEL IDC H. The CPU then will do a PORT.READ PORT ADRS[DATA.BYTE]. This instruction will be decoded by PORT INTERFACE REGISTER PAL and the

```

:19450 : PORT RD/WR DECODE PAL. The inputs to the PORT RD/WR DECODE PAL will
:19451 : be:
:19452 : CSR 17 H = H
:19453 : CSR 14 H = L
:19454 : CSR 13 H = L
:19455 : CSR 12 H = L
:19456 : CSR 11 H = H
:19457 : CSR 10 H = L
:19458 : PORT INSTR H = H
:19459 : CPU P2 H = H
:19460 : These inputs will assert READ DATA L and BYTE L.
:19461 : These two signals are inputs to the PORT USEQ A and B PALS. These PALS
:19462 : do not single-step with the CPU since they are clocked by CPU CLOCK H
:19463 : which is a free running clock.
:19464 : BRANCH TEST L will be asserted. When PORT USEQ PALS are clocked after
:19465 : READ DATA L and BYTE L assert, the outputs of PORT USEQ A will be:
:19466 : PORT NAD 0 H = H
:19467 : PORT NAD 1 H = H
:19468 : PORT NAD 2 H = H
:19469 : BRANCH TEST L = H
:19470 : PENB FIFO H = H
:19471 : PINCR FIFO CNTR H = H
:19472 : PWRITE FIFO H = L
:19473 :
:19474 : The only output of PORT USEQ B that will assert is PLOAD OUTREG B0 L.
:19475 :
:19476 : PENB FIFO H is ANDED with PSEL FIFOA H = H and asserts
:19477 : ENB FIFOA L which enables address 0 of FIFOA onto BUS OUT. PLOAD OUTREG
:19478 : B0 L clocks this data into the I/O BUS OUT LATCHES.
:19479 :
:19480 : At the same time, the PORT INTERFACE REG PAL is decoding the same
:19481 : instruction. The inputs to this PAL will be:
:19482 :
:19483 : CSR 17 H = H
:19484 : CSR 14 H = L
:19485 : READ PORT L = H
:19486 : CSR 12 H = L
:19487 : CSR 11 H = H
:19488 : CSR 10 H = L
:19489 : PORT INSTR H = H
:19490 : CPU P2 H = H
:19491 : On the next CPU CLOCK H, the outputs will be:
:19492 :
:19493 : SEL IDC H = H
:19494 : R2 L = H
:19495 : R1 L = L
:19496 : R0 L = H
:19497 : PSEL FIFOA H = H
:19498 :
:19499 : The CPU will then issue a MOV PORT TO LSC[] instruction. This will
:19500 : assert READ PORT L which is an input to the PORT INTERFACE REG PAL.
:19501 : At the next CPU CLOCK , output READ PORT H will assert.
:19502 : This signal is one of the enables to the decoder of R<2:0>.
:19503 : READ PORT H is also ANDED with SEL IDC H, which will be asserted,
:19504 : and SEL ACC L = H. This combination of inputs will assert READ IDC L

```

which is the enable to the decoder of R<2:0>.  
 The combination of R<2:0> inputs that were previously latched by the  
 PORT INTERFACE REG PAL, will assert SEL BYTE L. This signal is ORED  
 with SEL WORD L and will assert ENB OUTREG L. ENB OUTREG L is the  
 output enable for the I/O BUS OUT LATCHES and outputs the data onto  
 the I/O BUS.  
 READ IDC L which was previously asserted, directs the Tranceivers to  
 take the data from the I/O BUS and put it on the Y BUS. The CPU then  
 moves the data to an LS location.

# ERROR 2:

The same as error 1 except FIFO B will be selected.

FIFO B is selected by the port by a PORT.CONTROL [SEL.FIFO.B]  
 instruction. This instruction is decoded by the PORT INTERFACE REG PAL.  
 The inputs will be:

CSR 17 H = H  
 CSR 14 H = H  
 CSR 12 H = H  
 CSR 11 H = H  
 CSR 10 H = H  
 PORT INSTR H = H  
 CPU P2 H = H

When the PAL is clocked by CPU CLOCK H, PSEL FIFO A H will deassert.  
 PSEL FIFO B H = H, is the inverse of PSEL FIFO A H = L.

T.1D:

MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT 15 IN WR[0] FOR  
 MOV WR[0] TO LS[CONTROL.STATUS] ; CONTROL AND STATUS WORD  
 ;SET BIT 15 IN CONTROL AND  
 ; STATUS WORD - BIT 15  
 ; INDICATES START OF TEST TO  
 ; CONSOLE  
 ;ISSUE CPU ATTN TO CONSOLE  
 WAIT.T1D.0: ;LOOP HERE WAITING FOR  
 JMP [WAIT.T1D.0] ; CONSOLE TO RESPOND  
 CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER  
 ; IN LS  
 MOV LS[IDC] TO WR[0] ;STORE MODULE CODE IN LS TO  
 MOV WR[0] TO LS[MODULE.NUM] ; INDICATE IDC MODULE  
 MOV LS[#1] TO WR[0] ;SET WRO = 1  
 MOV WR[0] TO LS[ERROR.NUMBER] ;SET UP ERROR NUMBER = 1 IN LS  
 MOV LS[#FFFFFF00] TO WR[0] ;SET UP THE ERROR MASK TO  
 MOV WR[0] TO LS[ERROR.MASK] ; CHECK THE LOW BYTE

U 1133, B65E,15  
 U 1134, 3E80,15  
 U 1135, 10E0,15  
 U 1136, 8913,64  
 U 1137, 65A0,15  
 U 1138, B64A,15  
 U 1139, 3E8C,15  
 U 113A, B640,15  
 U 113B, BE82,15  
 U 113C, B62C,15  
 U 113D, 3E8A,15

G 16  
Fiche 2 Frame G16  
Sequence 407

ZZ-ENKCF-1.4 : ENKCF.MIC Test 1D DATA BUS IN AND DATA BUS OUT TEST (M8388 IDC MODULE)  
: ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40  
: ENKCF.MIC

U 113E, 2F85,15 :19560 CLR WR[2] ;USE AS A LOOP COUNTER  
:19561  
U 113F, B699,95 :19562 MOV LS[#AAAAAAAA] TO WR[3] ;GET THE DATA PATTERN  
:19563 CHK.NXT.PAT.T1D:  
U 1140, 452D,95 :19564 BIC LS[#FFFF00] TO WR[3] ;SAVE THE LOW BYTE (AA)  
:19565 LOOP.T1D.1:  
U 1141, A006,15 :19566 MOV WR[3] TO WR[0] ;MOV IT TO WR[0] FOR THE WRITE  
:19567  
U 1142, 17D8,15 :19568 PORT.CONTROL [SEL.FIFO.A] ;SELECT FIFO A  
U 1143, 17C0,15 :19569 PORT.CONTROL [CLEAR.FIFO.CNTR] ;CLEAR THE FIFO A ADDR CNTR  
U 1144, 97A8,15 :19570 PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;WR FIFOA ADDRESS 0 = AA  
U 1145, 17C0,15 :19571 PORT.CONTROL [CLEAR.FIFO.CNTR] ;SET ADDRS CNTR = 0  
:19572  
:19573 READ.FIFOA.T1D.1:  
U 1146, 9090,15 :19574 MISC [SET.PORT.I.0] ;SELECT THE IDC  
U 1147, 1788,15 :19575 PORT.READ PORT.ADRS[DATA.BYTE] ;SET UP TO READ FIFO A  
U 1148, 3B32,15 :19576 MOV PORT TO LS[PORT0] ;READ THE DATA FROM FIFO A  
:19577 ;SAVE IT IN LS  
U 1149, 3732,15 :19578 MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA  
U 114A, 2006,95 :19579 MOV WR[3] TO WR[1] ;SET UP EXPECTED DATA  
U 114B, 0869,3C :19580 JSR [CHECK.RESULT] ;CHECK THE DATA  
U 114C, 8914,14 :19581 JMP [LOOP.T1D.1] ;LOOP HERE IF ERR & LOE IS SET  
:19582  
:19583 CMP WR[2] WITH LS[ZERO], ;FIRST LOOP COUNT?  
U 114D, CF9D,35 :19584 DT(LONG)&SET.ALU.CC  
U 114E, 0915,21 :19585 JMP.IF[NEQ] TO [CHK.LPCNT.T1D.1] ;JMP IF NOT FIRST  
U 114F, 369B,95 :19586 MOV LS[#55555555] TO WR[3] ;GET DATA PATTERN 2  
U 1150, 2045,15 :19587 INC WR[2] ;INCR THE LOOP COUNT  
U 1151, 0914,04 :19588 JMP [CHK.NXT.PAT.T1D] ;CHECK THE NEXT PATTERN  
:19589  
:19590 CHK.LPCNT.T1D.1:  
:19591 CMP WR[2] WITH LS[#1], ;SECOND LOOP COUNT?  
U 1152, 4F41,35 :19592 DT(LONG)&SET.ALU.CC  
U 1153, 0915,71 :19593 JMP.IF[NEQ] TO [CHK.LPCNT.T1D.2]  
U 1154, B6C5,95 :19594 MOV LS[#33333333] TO WR[3] ;GET DATA PATTERN 3  
U 1155, 2045,15 :19595 INC WR[2] ;INCR THE LOOP COUNT  
U 1156, 0914,04 :19596 JMP [CHK.NXT.PAT.T1D] ;CHECK THE NEXT PATTERN  
:19597  
:19598 CHK.LPCNT.T1D.2:  
:19599 CMP WR[2] WITH LS[#2], ;THIRD LOOP COUNT?  
U 1157, CF43,35 :19600 DT(LONG)&SET.ALU.CC  
U 1158, 8915,C1 :19601 JMP.IF[NEQ] TO [CHECK.FIFOB.T1D] ;JMP IF DONE ALL PATS  
U 1159, 36C7,95 :19602 MOV LS[#0F0F0F0F] TO WR[3] ;GET DATA PATTERN 4  
U 115A, 2045,15 :19603 INC WR[2] ;INCR THE LOOP COUNT  
U 115B, 0914,04 :19604 JMP [CHK.NXT.PAT.T1D] ;CHECK THE NEXT PATTERN  
:19605  
:19606 ;TEST ONE LOCATION OF FIFO B  
:19607  
:19608 CHECK.FIFOB.T1D:  
U 115C, FF82,15 :19609 INC LS[ERROR.NUMBER] ;ERROR 2  
:19610  
U 115D, 2F85,15 :19611 CLR WR[2] ;USE AS A LOOP COUNTER  
:19612  
U 115E, B699,95 :19613 MOV LS[#AAAAAAAA] TO WR[3] ;GET THE DATA PATTERN  
:19614



```

:19615 CHK.NXT.PATB.T1D:
U 115F, 452D,95 :19616 BIC LS[FFFFFFF00] TO WR[3] ;SAVE THE LOW BYTE (AA)
:19617 LOOP.T1D.2:
U 1160, A006,15 :19618 MOV WR[3] TO WR[0] ;MOV PAT TO WR[0] FOR THE WRITE
:19619
U 1161, 97DC,15 :19620 PORT.CONTROL [SEL.FIFO.B] ;SELECT FIFO B
U 1162, 17C0,15 :19621 PORT.CONTROL [CLEAR.FIFO.CNTR] ;CLEAR THE FIFO B ADDR CNTR
U 1163, 97A8,15 :19622 PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;WR FIFO B ADDRESS 0 = AA
U 1164, 17C0,15 :19623 PORT.CONTROL [CLEAR.FIFO.CNTR] ;SET ADDRS CNTR = 0
:19624
:19625 READ.FIFOB.T1D.1:
U 1165, 9090,15 :19626 MISC [SET.PORT.I.0] ;SELECT THE IDC
U 1166, 1788,15 :19627 PORT.READ PORT.ADRS[DATA.BYTE] ;SET UP TO READ FIFO B
U 1167, 3B32,15 :19628 MOV PORT TO LS[PORT0] ;READ THE DATA FROM FIFO B
:19629 ;SAVE IT IN LS
U 1168, 3732,15 :19630 MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
U 1169, 2006,95 :19631 MOV WR[3] TO WR[1] ;SET UP EXPECTED DATA
U 116A, 0869,3C :19632 JSR [CHECK.RESULT] ;CHECK THE DATA
U 116B, 8916,04 :19633 JMP [LOOP.T1D.2] ;LOOP HERE IF ERR & LOE IS SET
:19634
:19635 ;FIRST LOOP COUNT?
U 116C, CF9D,35 :19636 CMP WR[2] WITH LS[ZERO],
U 116D, 8917,11 :19637 DT(LONG)&SET.ALU.CC
U 116E, 369B,95 :19638 JMP.IF[NEQ] TO [CHK.LPCNT.T1D.3] ;JMP IF NOT FIRST
U 116F, 2045,15 :19639 MOV LS[#55555555] TO WR[3] ;GET DATA PATTERN 2
U 1170, 8915,F4 :19640 INC WR[2] ;INCR THE LOOP COUNT
:19641 JMP [CHK.NXT.PATB.T1D] ;CHECK THE NEXT PATTERN
:19642
:19643 CHK.LPCNT.T1D.3:
U 1171, 4F41,35 :19644 CMP WR[2] WITH LS[#1],
U 1172, 0917,61 :19645 DT(LONG)&SET.ALU.CC
U 1173, B6C5,95 :19646 JMP.IF[NEQ] TO [CHK.LPCNT.T1D.4] ;SECOND LOOP COUNT?
U 1174, 2045,15 :19647 MOV LS[#33333333] TO WR[3] ;GET DATA PATTERN 3
U 1175, 8915,F4 :19648 INC WR[2] ;INCR THE LOOP COUNT
:19649 JMP [CHK.NXT.PATB.T1D] ;CHECK THE NEXT PATTERN
:19650
:19651 CHK.LPCNT.T1D.4:
U 1176, CF43,35 :19652 CMP WR[2] WITH LS[#2],
U 1177, 8917,B1 :19653 DT(LONG)&SET.ALU.CC
U 1178, 36C7,95 :19654 JMP.IF[NEQ] TO [CLEANUP.T1D] ;THIRD LOOP COUNT?
U 1179, 2045,15 :19655 MOV LS[#0F0F0F0F] TO WR[3] ;JMP IF DONE ALL PATS
U 117A, 8915,F4 :19656 INC WR[2] ;GET DATA PATTERN 4
:19657 JMP [CHK.NXT.PATB.T1D] ;INCR THE LOOP COUNT
:19658 ;CHECK THE NEXT PATTERN
:19659
:19660 CLEANUP.T1D:
U 117B, 8875,3C :19661 JSR [CLEANUP] ;CLEANUP CONDITIONS SET BY THIS
:19662 ;TEST
:19663
:19664 END.T1D: ;END TEST
:19665

```

```

:19666 .PAGE 'Test 1E FIFO A DATA INTEGRITY TEST (M8388 IDC MODULE)'
:19667 ++
:19668
:19669 .Test Description:
:19670
:19671 This test is designed to verify that every location in FIFO A can be
:19672 written and read correctly. This will accomplished by doing byte writes
:19673 to every location of FIFO A.
:19674 The data patterns that will be used are:
:19675
:19676 Data pattern 1 = All ones
:19677 Data Pattern 2 = All zeros
:19678 Data Pattern 3 = Float a one through a field of zeros.
:19679 Data Pattern 4 = Float a zero through a field of ones.
:19680
:19681 .Assumptions:
:19682
:19683 It is assumed that BUS IN and BUS OUT have been tested and found
:19684 fully operational.
:19685
:19686 .Test Steps:
:19687
:19688 1. Set up error mask, error number and module number in LS
:19689 (for error printouts) and clear the previous error number in LS.
:19690 2. Use WR[2] to track the address.
:19691 3. Clear WR[2].
:19692 4. Clear FIFO A Address Counter.
:19693 5. Move the Data Pattern from LS to WR[0].
:19694 6. Do a byte write to FIFO A.
:19695 7. Increment WR[2].
:19696 8. If WR[2] is less than 512, then go to Step 6.
:19697 9. Clear FIFO A Address Counter.
:19698 10. Read a byte from FIFO A.
:19699 11. Check it.
:19700 12. Increment WR[2].
:19701 13. If WR[2] is less than 512 go to Step 10.
:19702 14. Get the next data pattern and go to Step 3 until all data patterns
:19703 have been tested.
:19704
:19705 .Errors:
:19706
:19707 ERROR 1 - FIFO A failed.
:19708
:19709 Printout:
:19710 EXPECTED RECEIVED OTHER DATA
:19711 FIFO A FIFO A WR[2]
:19712 FIFOA ADDRESS
:19713
:19714 .Debug:
:19715 ERROR 1:
:19716
:19717 Because of hardware restrictions, this test will only loop on the
:19718 read of address 0 of FIFO A. This will allow verifying the enables.
:19719
:19720 The control logic and the data path has been verified in a previous

```

```

:19721 : test. Therefore, if the enables are OK, one must determine only which
:19722 : RAM contains the failure.
:19723 : FIFO A consists of 4 chips. Address 0 - 255 (0 - FF) are in two chips,
:19724 : and Address 256-511 (100 - 1FF) are in two chips.
:19725 : The failure can be isolated to which one of the two chips by examining
:19726 : the bit number that failed. Bits <3:0> are in one chip and
:19727 : bits <7:4> are in the other.
:19728 :
:19729 :--
:19730 :*****
:19731 :T.1E:
U 117C, B65E,15 :19732 :MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT 15 IN WR[0] FOR
:19733 :CONTROL AND STATUS WORD
U 117D, 3E80,15 :19734 :MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND
:19735 :STATUS WORD - BIT 15
:19736 :INDICATES START OF TEST TO
:19737 :CONSOLE
U 117E, 10E0,15 :19738 :MISC [SET.CP.ATTN] ;ISSUE CPU ATTN TO CONSOLE
:19739 :WAIT.T1E.0:
U 117F, 0917,F4 :19740 :JMP [WAIT.T1E.0] ;LOOP HERE WAITING FOR
:19741 :CONSOLE TO RESPOND
:19742 :
U 1180, 65A0,15 :19743 :CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
:19744 :IN LS
:19745 :
U 1181, B64A,15 :19746 :MOV LS[IDC] TO WR[0] ;STORE MODULE CODE IN LS TO
U 1182, 3E8C,15 :19747 :MOV WR[0] TO LS[MODULE.NUM] ;INDICATE IDC MODULE
:19748 :
:19749 :
U 1183, B640,15 :19750 :MOV LS[#1] TO WR[0] ;SET WRO = 1
U 1184, BE82,15 :19751 :MOV WR[0] TO LS[ERROR.NUMBER] ;SET UP ERROR NUMBER = 1 IN LS
:19752 :
:19753 :
U 1185, B62C,15 :19754 :MOV LS[#FFFFFF00] TO WR[0] ;SET UP THE ERROR MASK TO
U 1186, 3E8A,15 :19755 :MOV WR[0] TO LS[ERROR.MASK] ;CHECK THE LOW BYTE
:19756 :
:19757 :
U 1187, B670,15 :19758 :MOV LS[OTHER.DATA] TO WR[0] ;SET UP CONTROL.STATUS TO PRINT
U 1188, 3E80,15 :19759 :MOV WR[0] TO LS[CONTROL.STATUS] ;OTHER DATA IF ERROR
U 1189, DB00,15 :19760 :NOP
:19761 :CHECK FIFO WITH ALL ONES
:19762 :
U 118A, 2F85,15 :19763 :CLR WR[2] ;USE AS ADDRESS TRACKER
U 118B, 17D8,15 :19764 :PORT.CONTROL [SEL.FIFO.A] ;SELECT FIFO A FROM THE PORT
U 118C, 17C0,15 :19765 :PORT.CONTROL [CLEAR.FIFO.CNTR] ;SET FIFO A ADDRESS = 0
U 118D, B626,15 :19766 :MOV LS[#FF] TO WR[0] ;GET THE FIRST DATA PATTERN
:19767 :
:19768 :WRITE.FIFO.ONES.T1E:
U 118E, 97A8,15 :19769 :PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;WR BYTE OF DATA TO FIFO A
U 118F, 2045,15 :19770 :INC WR[2] ;INCR THE ADDRS TRACKER
:19771 :
:19772 :CMP WR[2] WITH LS[#200],
:19773 :DT(LONG)&SET.ALU.CC ;FIFO ALL WRITTEN WITH PAT?
U 1190, 4F53,35 :19773 :JMP.IF[NEQ] TO [WRITE.FIFO.ONES.T1E] ;JMP IF NOT DONE
:19774 :
:19775 :LOOP.T1E.PAT1:

```

K 16

ZZ-ENKCF-1.4 : ENKCF.MIC Test 1E FIFO A DATA INTEGRITY TEST (M8388 IDC MODULE)
 Fiche 2 Frame K16 Sequence 411

: ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40
 Page 405

: ENKCF.MIC Test 1E FIFO A DATA INTEGRITY TEST (M8388 IDC MODULE)

```

U 1192, 2F85,15 :19776 CLR WR[2] ;USE AS THE ADDRESS TRACKER
U 1193, 17C0,15 :19777 PORT.CONTROL [CLEAR.FIFO.CNTR] ;CLR THE ADDRESS COUNTER
 :19778
 :19779
U 1194, 9090,15 :19780 READ.FIFO.ONES.T1E:
U 1195, 1788,15 :19781 MISC [SET.PORT.I.0] ;SELECT THE IDC
U 1196, 3B32,15 :19782 PORT.READ PORT.ADRS[DATA.BYTE] ;SET UP TO READ THE FIFO
 :19783 MOV PORT TO LS[PORT0] ;READ A BYTE OF DATA FROM THE
 :19784 MOV WR[2] TO LS[ADDRESS.DATA] ;FIFO AND SAVE IT IN LS
 :19785 MOV LS[PORT0] TO WR[0] ;ADDRESS COUNTER
U 1197, 3E89,15 :19786 MOV LS[#FF] TO WR[1] ;SET UP RECEIVED DATA
U 1198, 0869,3C :19787 JSR [CHECK.RESULT] ;SET UP EXPECTED DATA
U 1198, 0919,24 :19788 JMP [LOOP.T1E.PAT1] ;CHECK THE DATA
U 119C, 2045,15 :19789 INC WR[2] ;LOOP HERE IF ERR & LOE IS SET
 :19790 CMP WR[2] WITH LS[#200], ;INCREMENT THE ADRS CNTR
 :19791 DT(LONG)&SET.ALU.CC
U 119D, 4F53,35 :19792 JMP.IF[NEQ] TO [READ.FIFO.ONES.T1E] ;CHECKED ALL LOCATIONS?
U 119E, 0919,41 :19793 ;CHECK FIFO WITH ALL ZEROS
 :19794
 :19795
U 119F, 2F85,15 :19796 CLR WR[2] ;USE AS ADDRESS TRACKER
U 11A0, 17D8,15 :19797 PORT.CONTROL [SEL.FIFO.A] ;SELECT FIFO A FROM THE PORT
U 11A1, 17C0,15 :19798 PORT.CONTROL [CLEAR.FIFO.CNTR] ;SET FIFO A ADDRESS = 0
U 11A2, 2F80,15 :19799 CLR WR[0] ;GET THE DATA PATTERN
 :19800
 :19801
U 11A3, 97A8,15 :19802 WRITE.FIFO.ZERO.T1E:
U 11A4, 2045,15 :19803 PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;WR A BYTE OF DATA TO FIFO A
 :19804 INC WR[2] ;INCR THE ADRS TRACKER
 :19805 CMP WR[2] WITH LS[#200],
U 11A5, 4F53,35 :19806 DT(LONG)&SET.ALU.CC ;FIFO ALL WRITTEN WITH PAT?
U 11A6, 891A,31 :19807 JMP.IF[NEQ] TO [WRITE.FIFO.ZERO.T1E] ;JMP IF NOT DONE
 :19808
 :19809
U 11A7, 2F85,15 :19810 LOOP.T1E.PAT2:
U 11A8, 17C0,15 :19811 CLR WR[2] ;USE AS THE ADDRESS TRACKER
 :19812 PORT.CONTROL [CLEAR.FIFO.CNTR] ;CLR THE ADDRESS COUNTER
 :19813
 :19814
U 11A9, 9090,15 :19815 READ.FIFO.ZERO.T1E:
U 11AA, 1788,15 :19816 MISC [SET.PORT.I.0] ;SELECT THE IDC
U 11AB, 3B32,15 :19817 PORT.READ PORT.ADRS[DATA.BYTE] ;SET UP TO READ THE FIFO
 :19818 MOV PORT TO LS[PORT0] ;READ A BYTE OF DATA FROM THE
 :19819 MOV WR[2] TO LS[ADDRESS.DATA] ;FIFO AND SAVE IT IN LS
 :19820 MOV LS[PORT0] TO WR[0] ;ADDRESS COUNTER
U 11AC, 3E89,15 :19821 CLR WR[1] ;SET UP RECEIVED DATA
U 11AD, 3732,15 :19822 JSR [CHECK.RESULT] ;SET UP EXPECTED DATA
U 11AE, 2F82,95 :19823 JMP [LOOP.T1E.PAT2] ;CHECK THE DATA
U 11AF, 0869,3C :19824 INC WR[2] ;LOOP HERE IF ERR & LOE IS SET
U 11B0, 091A,74 :19825 CMP WR[2] WITH LS[#200], ;INCREMENT THE ADRS CNTR
U 11B1, 2045,15 :19826 DT(LONG)&SET.ALU.CC ;CHECKED ALL LOCATIONS?
 :19827 JMP.IF[NEQ] TO [READ.FIFO.ZERO.T1E] ;JMP IF NOT DONE
 :19828
 :19829
U 11B2, 4F53,35 :19830 ;FLOATINGS ONES PATTERN
U 11B3, 891A,91 :19831
 :19832
U 11B4, 17D8,15 :19833 PORT.CONTROL [SEL.FIFO.A] ;SELECT FIFO A FROM THE PORT
U 11B5, E5F8,15 :19834 CLR LS[05] ;CLR THE INDEX TO THE DATA PAT

```



```

U 11D6, 4F53,35 :19886 DT(LONG)&SET.ALU.CC ;FIFO ALL WRITTEN WITH PAT?
U 11D7, 091D,31 :19887 JMP.IF[NEQ] TO [WRITE.FIFO.FLOATO.T1E] ;JMP IF NOT DONE
:19888
:19889 LOOP.T1E.PAT4:
U 11D8, 2F85,15 :19890 CLR WR[2] ;USE AS THE ADDRESS TRACKER
U 11D9, 17C0,15 :19891 PORT.CONTROL [CLEAR.FIFO.CNTR] ;CLR THE ADDRESS COUNTER
:19892
:19893 READ.FIFO.FLOATO.T1E:
U 11DA, 9090,15 :19894 MISC [SET.PORT.I.0] ;SELECT THE IDC
U 11DB, 1788,15 :19895 PORT.READ PORT.ADRS[DATA.BYTE] ;SET UP TO READ THE FIFO
U 11DC, 3B32,15 :19896 MOV PORT TO LS[PORT0] ;READ A BYTE OF DATA FROM THE
:19897 ; FIFO AND SAVE IT IN LS
U 11DD, 3E89,15 :19898 MOV WR[2] TO LS[ADDRESS.DATA] ;ADDRESS COUNTER
U 11DE, 3732,15 :19899 MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
U 11DF, B6F6,95 :19900 MOV LS[SHIFT.OS(4-0)] TO WR[1] ;SET UP EXPECTED DATA
U 11E0, 0869,3C :19901 JSR [CHECK.RESULT] ;CHECK THE DATA
U 11E1, 891D,84 :19902 JMP [LOOP.T1E.PAT4] ;LOOP HERE IF ERR & LOE IS SET
U 11E2, 2045,15 :19903 INC WR[2] ;INCREMENT THE ADRS CNTR
:19904 CMP WR[2] WITH LS[#200], ;CHECKED ALL LOCATIONS WITH
U 11E3, 4F53,35 :19905 DT(LONG)&SET.ALU.CC ; THIS DATA PATTERN?
U 11E4, 091D,A1 :19906 JMP.IF[NEQ] TO [READ.FIFO.FLOATO.T1E] ;JMP IF NOT DONE
:19907
U 11E5, 7FF8,15 :19908 INC LS[OS] ;INCR THE INDEX TO THE NEXT
:19909 ; DATA PATTERN
:19910 CMP LS[OS] WITH WR[3], ;DONE WITH THE FLOATING ONES
U 11E6, 4EF9,B5 :19911 DT(LONG)&SET.ALU.CC ; DATA PATTERNS?
U 11E7, 891D,11 :19912 JMP.IF[NEQ] TO [WRITE.NEXT.OPAT.T1E] ;JMP IF NOT DONE
:19913
U 11E8, 8875,3C :19914 JSR [CLEANUP] ;CLEANUP CONDITIONS SET BY THIS
:19915 ; TEST
:19916
:19917 END.T1E: ;END TEST
:19918
:19919

```

U  
U

```

:19975 : test. Therefore, if the enables are OK, one must determine only which
:19976 : RAM contains the failure.
:19977 : FIFO B consists of 4 chips. Address 0 - 255 (0 - FF) are in two chips,
:19978 : and Address 256-511 (100 - 1FF) are in two chips.
:19979 : The failure can be isolated to which one of the two chips by examining
:19980 : the bit number that failed. Bits <3:0> are in one chip and
:19981 : bits <7:4> are in the other.
:19982 :
:19983 :--
:19984 :*****
:19985 :T.1F:
U 11E9, B65E,15 :19986 :MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT 15 IN WR[0] FOR
:19987 : ;CONTROL AND STATUS WORD
U 11EA, 3E80,15 :19988 :MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND
:19989 : ;STATUS WORD - BIT 15
:19990 : ;INDICATES START OF TEST TO
:19991 : ;CONSOLE
U 11EB, 10E0,15 :19992 :MISC [SET.CP.ATTN] ;ISSUE CPU ATTN TO CONSOLE
:19993 :WAIT.T1F.0:
U 11EC, 091E,C4 :19994 :JMP [WAIT.T1F.0] ;LOOP HERE WAITING FOR
:19995 : ;CONSOLE TO RESPOND
:19996 :
U 11ED, 65A0,15 :19997 :CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
:19998 : ;IN LS
:19999 :
U 11EE, B64A,15 :20000 :MOV LS[IDC] TO WR[0] ;STORE MODULE CODE IN LS TO
U 11EF, 3E8C,15 :20001 :MOV WR[0] TO LS[MODULE.NUM] ;INDICATE IDC MODULE
:20002 :
:20003 :
U 11F0, B640,15 :20004 :MOV LS[#1] TO WR[0] ;SET WR0 = 1
U 11F1, BE82,15 :20005 :MOV WR[0] TO LS[ERROR.NUMBER] ;SET UP ERROR NUMBER = 1 IN LS
:20006 :
:20007 :
U 11F2, B62C,15 :20008 :MOV LS[FFFFFFFF00] TO WR[0] ;SET UP THE ERROR MASK TO
U 11F3, 3E8A,15 :20009 :MOV WR[0] TO LS[ERROR.MASK] ;CHECK THE LOW BYTE
:20010 :
:20011 :
U 11F4, B670,15 :20012 :MOV LS[OTHER.DATA] TO WR[0] ;SET UP CONTROL.STATUS TO PRINT
U 11F5, 3E80,15 :20013 :MOV WR[0] TO LS[CONTROL.STATUS] ;OTHER DATA IF ERROR
:20014 :
:20015 :;CHECK FIFO WITH ALL ONES
:20016 :
U 11F6, 2F85,15 :20017 :CLR WR[2] ;USE AS ADDRESS TRACKER
U 11F7, 97DC,15 :20018 :PORT.CONTROL [SEL.FIFO.B] ;SELECT FIFO B FROM THE PORT
U 11F8, 17C0,15 :20019 :PORT.CONTROL [CLEAR.FIFO.CNTR] ;SET FIFO B ADDRESS = 0
U 11F9, B626,15 :20020 :MOV LS[FF] TO WR[0] ;GET THE FIRST DATA PATTERN
:20021 :
:20022 :WRITE.FIFO.ONES.T1F:
U 11FA, 97A8,15 :20023 :PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;WR BYTE OF DATA TO FIFO B
U 11FB, 2045,15 :20024 :INC WR[2] ;INCR THE ADDRS TRACKER
:20025 :
:20026 :CMP WR[2] WITH LS[#200],
U 11FC, 4F53,35 :20026 :DT(LONG)&SET.ALU.CC ;FIFO ALL WRITTEN WITH PAT?
U 11FD, 891F,A1 :20027 :JMP.IF[NEQ] TO [WRITE.FIFO.ONES.T1F] ;JMP IF NOT DONE
:20028 :
:20029 :LOOP.T1F.PAT1:

```



E 1

ZZ-ENKCF-1.4 : ENKCF.MIC Test 1F FIFO B DATA INTEGRITY TEST (M8388 IDC MODULE)
 Fiche 3 Frame E1 Sequence 416

: ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40
 Page 410

: ENKCF.MIC Test 1F FIFO B DATA INTEGRITY TEST (M8388 IDC MODULE)

|                 |        |                                            |                                |
|-----------------|--------|--------------------------------------------|--------------------------------|
| U 11FE, 2F85,15 | :20030 | CLR WR[2]                                  | :USE AS THE ADDRESS TRACKER    |
| U 11FF, 17C0,15 | :20031 | PORT.CONTROL [CLEAR.FIFO.CNTR]             | :CLR THE ADDRESS COUNTER       |
|                 | :20032 |                                            |                                |
|                 | :20033 | READ.FIFO.ONES.T1F:                        |                                |
| U 1200, 9090,15 | :20034 | MISC [SET.PORT.I.O]                        | :SELECT THE IDC                |
| U 1201, 1788,15 | :20035 | PORT.READ PORT.ADRS[DATA.BYTE]             | :SET UP TO READ THE FIFO       |
| U 1202, 3B32,15 | :20036 | MOV PORT TO LS[PORT0]                      | :READ A BYTE OF DATA FROM THE  |
|                 | :20037 |                                            | : FIFO AND SAVE IT IN LS       |
| U 1203, 3E89,15 | :20038 | MOV WR[2] TO LS[ADDRESS.DATA]              | :ADDRESS COUNTER               |
| U 1204, 3732,15 | :20039 | MOV LS[PORT0] TO WR[0]                     | :SET UP RECEIVED DATA          |
| U 1205, 3626,95 | :20040 | MOV LS[#FF] TO WR[1]                       | :SET UP EXPECTED DATA          |
| U 1206, 0869,3C | :20041 | JSR [CHECK.RESULT]                         | :CHECK THE DATA                |
| U 1207, 091F,E4 | :20042 | JMP [LOOP.T1F.PAT1]                        | :LOOP HERE IF ERR & LOE IS SET |
| U 1208, 2045,15 | :20043 | INC WR[2]                                  | :INCREMENT THE ADRS CNTR       |
|                 | :20044 | CMP WR[2] WITH LS[#200],                   |                                |
| U 1209, 4F53,35 | :20045 | DT(LONG)&SET.ALU.CC                        | :CHECKED ALL LOCATIONS?        |
| U 120A, 8920,01 | :20046 | JMP.IF[NEQ] TO [READ.FIFO.ONES.T1F]        | :JMP IF NOT DONE               |
|                 | :20047 |                                            |                                |
|                 | :20048 | :CHECK FIFO WITH ALL ZEROS                 |                                |
|                 | :20049 |                                            |                                |
| U 120B, 2F85,15 | :20050 | CLR WR[2]                                  | :USE AS ADDRESS TRACKER        |
| U 120C, 97DC,15 | :20051 | PORT.CONTROL [SEL.FIFO.B]                  | :SELECT FIFO B FROM THE PORT   |
| U 120D, 17C0,15 | :20052 | PORT.CONTROL [CLEAR.FIFO.CNTR]             | :SET FIFO B ADDRESS = 0        |
| U 120E, 2F80,15 | :20053 | CLR WR[0]                                  | :GET THE DATA PATTERN          |
|                 | :20054 |                                            |                                |
|                 | :20055 | WRITE.FIFO.ZERO.T1F:                       |                                |
| U 120F, 97A8,15 | :20056 | PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] | :WR BYTE OF DATA TO FIFO B     |
| U 1210, 2045,15 | :20057 | INC WR[2]                                  | :INCR THE ADRS TRACKER         |
|                 | :20058 | CMP WR[2] WITH LS[#200],                   |                                |
| U 1211, 4F53,35 | :20059 | DT(LONG)&SET.ALU.CC                        | :FIFO ALL WRITTEN WITH PAT?    |
| U 1212, 8920,F1 | :20060 | JMP.IF[NEQ] TO [WRITE.FIFO.ZERO.T1F]       | :JMP IF NOT DONE               |
|                 | :20061 |                                            |                                |
|                 | :20062 | LOOP.T1F.PAT2:                             |                                |
| U 1213, 2F85,15 | :20063 | CLR WR[2]                                  | :USE AS THE ADDRESS TRACKER    |
| U 1214, 17C0,15 | :20064 | PORT.CONTROL [CLEAR.FIFO.CNTR]             | :CLR THE ADDRESS COUNTER       |
|                 | :20065 |                                            |                                |
|                 | :20066 | READ.FIFO.ZERO.T1F:                        |                                |
| U 1215, 9090,15 | :20067 | MISC [SET.PORT.I.O]                        | :SELECT THE IDC                |
| U 1216, 1788,15 | :20068 | PORT.READ PORT.ADRS[DATA.BYTE]             | :SET UP TO READ THE FIFO       |
| U 1217, 3B32,15 | :20069 | MOV PORT TO LS[PORT0]                      | :READ A BYTE OF DATA FROM THE  |
|                 | :20070 |                                            | : FIFO AND SAVE IT IN LS       |
| U 1218, 3E89,15 | :20071 | MOV WR[2] TO LS[ADDRESS.DATA]              | :ADDRESS COUNTER               |
| U 1219, 3732,15 | :20072 | MOV LS[PORT0] TO WR[0]                     | :SET UP RECEIVED DATA          |
| U 121A, 2F82,95 | :20073 | CLR WR[1]                                  | :SET UP EXPECTED DATA          |
| U 121B, 0869,3C | :20074 | JSR [CHECK.RESULT]                         | :CHECK THE DATA                |
| U 121C, 0921,34 | :20075 | JMP [LOOP.T1F.PAT2]                        | :LOOP HERE IF ERR & LOE IS SET |
| U 121D, 2045,15 | :20076 | INC WR[2]                                  | :INCREMENT THE ADRS CNTR       |
|                 | :20077 | CMP WR[2] WITH LS[#200],                   |                                |
| U 121E, 4F53,35 | :20078 | DT(LONG)&SET.ALU.CC                        | :CHECKED ALL LOCATIONS?        |
| U 121F, 0921,51 | :20079 | JMP.IF[NEQ] TO [READ.FIFO.ZERO.T1F]        | :JMP IF NOT DONE               |
|                 | :20080 |                                            |                                |
|                 | :20081 | :FLOATINGS ONES PATTERN                    |                                |
|                 | :20082 |                                            |                                |
| U 1220, 97DC,15 | :20083 | PORT.CONTROL [SEL.FIFO.B]                  | :SELECT FIFO B FROM THE PORT   |
| U 1221, E5F8,15 | :20084 | CLR LS[05]                                 | :CLR THE INDEX TO THE DATA PAT |

```

F-1
ZZ-ENKCF-1.4 : ENKCF.MIC Test 1F FIFO B DATA INTEGRITY TEST (M8388 IDC MODULE) Fiche 3 Frame F1 Sequence 417 Page 411
: ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40
: ENKCF.MIC Test 1F FIFO B DATA INTEGRITY TEST (M8388 IDC MODULE)

U 1222, B647,95 :20085 MOV LS[#8] TO WR[3] ;LAST INDEX TO WRITE A BYTE
:20086 ; WITH FLOATING PAT
:20087
:20088 WRITE.NEXT.1PAT.T1F:
U 1223, 17C0,15 :20089 PORT.CONTROL [CLEAR.FIFO.CNTR] ;SET FIFO B ADDRESS = 0
U 1224, 2F85,15 :20090 CLR WR[2] ;USE AS ADDRESS TRACKER
:20091
:20092 WRITE.FIFO.FLOAT1.T1F:
U 1225, 36F6,15 :20093 MOV LS[SHIFT.OS(4-0)] TO WR[0] ;PUT THE DATA PAT IN WR[0]
U 1226, 97A8,15 :20094 PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;WR BYTE OF DATA TO FIFO B
U 1227, 2045,15 :20095 INC WR[2] ;INCR THE ADPRS TRACKER
:20096 CMP WR[2] WITH LS[#200],
U 1228, 4F53,35 :20097 DT(LONG)&SET.ALU.CC ;FIFO ALL WRITTEN WITH PAT?
U 1229, 0922,51 :20098 JMP.IF[NEQ] TO [WRITE.FIFO.FLOAT1.T1F] ;JMP IF NOT DONE
:20099
:20100 LOOP.T1F.PAT3:
U 122A, 2F85,15 :20101 CLR WR[2] ;USE AS THE ADDRESS TRACKER
U 122B, 17C0,15 :20102 PORT.CONTROL [CLEAR.FIFO.CNTR] ;CLR THE ADDRESS COUNTER
:20103
:20104 READ.FIFO.FLOAT1.T1F:
U 122C, 9090,15 :20105 MISC [SET.PORT.1.0] ;SELECT THE IDC
U 122D, 1788,15 :20106 PORT.READ PORT.ADRS[DATA.BYTE] ;SET UP TO READ THE FIFO
U 122E, 3B32,15 :20107 MOV PORT TO LS[PORT0] ;READ A BYTE OF DATA FROM THE
:20108 ; FIFO AND SAVE IT IN LS
U 122F, 3E89,15 :20109 MOV WR[2] TO LS[ADDRESS.DATA] ;ADDRESS COUNTER
U 1230, 3732,15 :20110 MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
U 1231, B6F6,95 :20111 MOV LS[SHIFT.OS(4-0)] TO WR[1] ;SET UP EXPECTED DATA
U 1232, 0869,3C :20112 JSR [CHECK.RESULT] ;CHECK THE DATA
U 1233, 0922,A4 :20113 JMP [LOOP.T1F.PAT3] ;LOOP HERE IF ERR & LOE IS SET
U 1234, 2045,15 :20114 INC WR[2] ;INCREMENT THE ADPRS CNTR
:20115 CMP WR[2] WITH LS[#200], ;CHECKED ALL LOCATIONS WITH
U 1235, 4F53,35 :20116 DT(LONG)&SET.ALU.CC ; THIS DATA PATTERN?
U 1236, 0922,C1 :20117 JMP.IF[NEQ] TO [READ.FIFO.FLOAT1.T1F] ;JMP IF NOT DONE
:20118
U 1237, 7FF8,15 :20119 INC LS[05] ;INCR THE INDEX TO THE NEXT
:20120 ; DATA PATTERN
:20121 CMP LS[05] WITH WR[3], ;DONE WITH THE FLOATING ONES
U 1238, 4EF9,B5 :20122 DT(LONG)&SET.ALU.CC ; DATA PATTERNS?
U 1239, 0922,31 :20123 JMP.IF[NEQ] TO [WRITE.NEXT.1PAT.T1F] ;JMP IF NOT DONE
:20124
:20125 ;FLOATING ZERO PATTERN
:20126
U 123A, 97DC,15 :20127 PORT.CONTROL [SEL.FIFO.B] ;SELECT FIFO B FROM THE PORT
U 123B, E5F8,15 :20128 CLR LS[05] ;CLR THE INDEX TO THE DATA PAT
U 123C, B647,95 :20129 MOV LS[#8] TO WR[3] ;LAST INDEX TO WRITE A BYTE
:20130 ; WITH FLOATING PAT
:20131 WRITE.NEXT.OPAT.T1F:
U 123D, 17C0,15 :20132 PORT.CONTROL [CLEAR.FIFO.CNTR] ;SET FIFO B ADDRESS = 0
U 123E, 2F85,15 :20133 CLR WR[2] ;USE AS ADDRESS TRACKER
:20134
:20135 WRITE.FIFO.FLOAT0.T1F:
U 123F, 36F6,15 :20136 MOV LS[SHIFT.OS(4-0)] TO WR[0] ;PUT THE DATA PAT IN WR[0]
U 1240, 97A8,15 :20137 PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;WR BYTE OF DATA TO FIFO B
U 1241, 2045,15 :20138 INC WR[2] ;INCR THE ADPRS TRACKER
:20139 CMP WR[2] WITH LS[#200],

```

```

U 1242, 4F53,35 :20140 DT(LONG)&SET.ALU.CC ;FIFO ALL WRITTEN WITH PAT?
U 1243, 8923,F1 :20141 JMP.IF[NEQ] TO [WRITE.FIFO.FLOAT0.T1F] ;JMP IF NOT DONE
:20142
:20143 LOOP.T1F.PAT4.
U 1244, 2F85,15 :20144 CLR WR[2] ;USE AS THE ADDRESS TRACKER
U 1245, 17C0,15 :20145 PORT.CONTROL [CLEAR.FIFO.CNTR] ;CLR THE ADDRESS COUNTER
:20146
:20147 READ.FIFO.FLOAT0.T1F:
U 1246, 9090,15 :20148 MISC [SET.PORT.I.0] ;SELECT THE IDC
U 1247, 1788,15 :20149 PORT.READ PORT.ADRS[DATA.BYTE] ;SET UP TO READ THE FIFO
U 1248, 3B32,15 :20150 MOV PORT TO LS[PORT0] ;READ A BYTE OF DATA FROM THE
:20151 ; FIFO AND SAVE IT IN LS
U 1249, 3E89,15 :20152 MOV WR[2] TO LS[ADDRESS.DATA] ;ADDRESS COUNTER
U 124A, 3732,15 :20153 MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
U 124B, 86F6,95 :20154 MOV LS[SHIFT.OS(4-0)] TO WR[1] ;SET UP EXPECTED DATA
U 124C, 0869,3C :20155 JSR [CHECK.RESULT] ;CHECK THE DATA
U 124D, 8924,44 :20156 JMP [LOOP.T1F.PAT4] ;LOOP HERE IF ERR & LOE IS SET
U 124E, 2045,15 :20157 INC WR[2] ;INCREMENT THE ADRS CNTR
:20158 CMP WR[2] WITH LS[#200], ;CHECKED ALL LOCATIONS WITH
U 124F, 4F53,35 :20159 DT(LONG)&SET.ALU.CC ; THIS DATA PATTERN?
U 1250, 0924,61 :20160 JMP.IF[NEQ] TO [READ.FIFO.FLOAT0.T1F] ;JMP IF NOT DONE
:20161
U 1251, 7FF8,15 :20162 INC LS[OS] ;INCR THE INDEX TO THE NEXT
:20163 ; DATA PATTERN
:20164 CMP LS[OS] WITH WR[3], ;DONE WITH THE FLOATING ONES
U 1252, 4EF9,B5 :20165 DT(LONG)&SET.ALU.CC ; DATA PATTERNS?
U 1253, 0923,D1 :20166 JMP.IF[NEQ] TO [WRITE.NEXT.OPAT.T1F] ;JMP IF NOT DONE
:20167
U 1254, 8875,3C :20168 JSR [CLEANUP] ;CLEANUP CONDITIONS SET BY THIS
:20169 ; TEST
:20170
:20171 END.T1F: ;END TEST
:20172
:20173
:20174

```

:20175 :PAGE "Test 20 I/O BUS IN AND OUT LATCHES TEST (M8388 IDC MODULE)"

:20176 :++

:20177 :  
:20178 :Test Description:

:20179 :  
:20180 :This test is designed to verify the I/O BUS IN and I/O BUS OUT LATCHES  
:20181 :which buffer data to and from the FIFOs. This will be accomplished by  
:20182 :doing longword writes to FIFO A and then doing longword reads of  
:20183 :the data. The data patterns that will be used are:

:20184 :  
:20185 :Data pattern 1 = AAAAAAAA  
:20186 :Data pattern 2 = 55555555  
:20187 :Data pattern 3 = 33333333  
:20188 :Data pattern 4 = 0F0F0F0F  
:20189 :Data pattern 5 = 00FF00FF  
:20190 :Data pattern 6 = 0000FFFF

:20191 :  
:20192 :Assumptions:

:20193 :  
:20194 :It is assumed that all previous tests have run successfully.

:20195 :  
:20196 :Test Steps:

- :20197 :  
:20198 :1. Set up error mask, error number and module number in LS  
:20199 : (for error printouts) and clear the previous error number in LS.  
:20200 :2. Mov the data pattern from LS to WR[0].  
:20201 :3. Clear the FIFO Address Counter.  
:20202 :4. Do a WRITE.DATA.WORD to FIFO A. ( Word = 32 bits.)  
:20203 :5. Clear the FIFO Address Counter.  
:20204 :6. Read FIFO A.  
:20205 :7. Check the data.  
:20206 :8. Get the next data pattern and go to Step 2 until all patterns have  
:20207 : been tested.

:20208 :  
:20209 :Errors:

:20210 :  
:20211 :ERROR 1 - The data read from FIFO A was incorrect.

:20212 :  
:20213 :Printout:

:20214 :EXPECTED RECEIVED  
:20215 :FIFO A DATA FIFO A DATA

:20216 :Debug:

:20217 :  
:20218 :ERROR 1:

:20219 :  
:20220 :FIFO A is selected by the port by a PORT.CONTROL [SEL.FIFO.A]  
:20221 :instruction. This instruction is decoded by the PORT INTERFACE REG PAL.  
:20222 :The inputs will be:

:20223 :CSR 17 H = H  
:20224 :CSR 14 H = H  
:20225 :CSR 12 H = H  
:20226 :CSR 11 H = H  
:20227 :CSR 10 H = L  
:20228 :PORT INSTR H = H  
:20229 :CPU P2 H = H

:20230 : When the PAL is clocked by CPU CLOCK H, PSEL FIFO A H will assert.

:20231 :

:20232 :

:20233 :

:20234 :

:20235 :

:20236 :

:20237 :

:20238 :

:20239 :

:20240 :

:20241 :

:20242 :

:20243 :

:20244 :

:20245 :

:20246 :

:20247 :

:20248 :

:20249 :

:20250 :

:20251 :

:20252 :

:20253 :

:20254 :

:20255 :

:20256 :

:20257 :

:20258 :

:20259 :

:20260 :

:20261 :

:20262 :

:20263 :

:20264 :

:20265 :

:20266 :

:20267 :

:20268 :

:20269 :

:20270 :

:20271 :

:20272 :

:20273 :

:20274 :

:20275 :

:20276 :

:20277 :

:20278 :

:20279 :

:20280 :

:20281 :

:20282 :

:20283 :

:20284 :

WRITE.DATA.WORD:

This instruction will be decoded by the PORT RD/WR DECODE PAL.  
When the instruction is issued, the inputs to this PAL will be:

CSR 17 H = H  
CSR 14 H = L  
CSR 13 H = H  
CSR 12 H = L  
CSR 11 H = H  
CSR 10 H = H  
PORT INSTR H = H  
CPU P2 H = H

This combination of inputs will assert WRITE DATA L and STROBE  
DATA H. STROBE DATA H will clock the data from the I/O BUS into the  
I/O BUS IN LATCHES.

WRITE DATA L is an input to the PORT USEQ A and B PALS.  
These PALS cannot be single stepped. If the CPU is in single step  
mode when this instruction is issued, the PORT USEQ A and B PALS  
will complete the following sequence before the next micro-state.

On the first CPU CLOCK H pulse after WRITE DATA L asserts, the PORT  
USEQ PALS will enable Byte 0 to be written into  
FIFO A and then the address is incremented. On the next clock pulse  
the next byte is written and the address incremented. This continues  
until the entire longword has been written into the FIFO. At the end  
of this sequence, BRANCH TEST L will assert to wait for the next  
instruction.

READ.DATA.WORD:

The CPU will clear the FIFO Address Counter and then read the FIFO.  
The CPU must then issue a MISC [SET.PORT.I.O] which will assert  
SEL IDC H. The CPU will then issue the PORT.READ PORT ADRS[DATA.WORD]  
instruction. This instruction will be decoded by the PORT INTERFACE REG  
PAL and the PORT RD/WR DECODE PAL. The inputs to the PORT RD/WR DECODE  
PAL will be:

CSR 17 H = H  
CSR 14 H = L  
CSR 13 H = L  
CSR 12 H = L  
CSR 11 H = H  
CSR 10 H = H  
PORT INSTR H = H  
CPU P2 H = H

These inputs will assert READ DATA L. This signal as an input to the  
PORT USEQ A PAL when BRANCH TEST L is asserted and the PAL is clocked,  
will assert PENB FIFO H and PINCR FIFO CNTR H. PENB FIFO H is ANDED  
with PSEL FIFOA H and will assert ENB FIFOA L which enables the data  
from the FIFO onto BUS OUT. At the same time, READ DATA L as an input  
to PORT USEQ B PAL will assert PLOAD OUTREG B0 L when BRANCH TEST L  
is asserted and the PAL is clocked. PLOAD OUTREG B0 L will clock the  
data on BUS OUT into Byte 0 of the I/O BUS OUT LATCHES. The FIFO

U 12  
U 12

U 12

U 12  
U 12

U 12  
U 12

U 12

U 12  
U 12

U 12  
U 12

U 12  
U 12

U 12  
U 12

U 12  
U 12

U 12  
U 12

U 12

U 12

:20285 : Address Counter will be clocked by PINCR FIFO CNTR H. The PORT USED  
 :20286 : PALS will then sequence through and output the data from the FIFO to  
 :20287 : the I/O BUS OUT LATCHES a byte at a time. When the last byte has been  
 :20288 : loaded, BRANCH TEST L will assert to wait for the next instruction.  
 :20289 : The CPU will issue a MOV PORT TO LS[] instruction. This will assert  
 :20290 : READ PORT L which is an input to the PORT INTERFACE REG PAL. At the  
 :20291 : next CPU clock pulse, output READ PORT H will assert. This signal  
 :20292 : will be AND'ed with SEL IDC H and SEL ACC L and will assert READ  
 :20293 : IDC L. READ PORT H and READ IDC L are the enables for the Decoder of  
 :20294 : R<2:0> which were outputs from the PORT INTERFACE REG PAL. When R<2:0>  
 :20295 : are decoded, SEL WORD L will assert. This signal will assert ENB  
 :20296 : OUTREG L which will enable the I/O BUS OUT LATCHES onto the I/O BUS.  
 :20297 : READ IDC L as the direction input to the Y BUS Transceivers,  
 :20298 : will direct the I/O BUS onto the Y BUS.  
 :20299 : --

:20300 : \*\*\*\*\*

|                 |        |                  |                                            |                                 |
|-----------------|--------|------------------|--------------------------------------------|---------------------------------|
| U 1255, B65E,15 | :20301 | T.20:            | MOV LS[BEGIN.TEST] TO WR[0]                | :SET BIT 15 IN WR[0] FOR        |
|                 | :20302 |                  |                                            | : CONTROL AND STATUS WORD       |
| U 1256, 3E80,15 | :20303 |                  | MOV WR[0] TO LS[CONTROL.STATUS]            | :SET BIT 15 IN CONTROL AND      |
|                 | :20304 |                  |                                            | : STATUS WORD - BIT 15          |
|                 | :20305 |                  |                                            | : INDICATES START OF TEST TO    |
|                 | :20306 |                  |                                            | : CONSOLE                       |
| U 1257, 10E0,15 | :20307 |                  | MISC [SET.CP.ATTN]                         | :ISSUE CPU ATTN TO CONSOLE      |
|                 | :20308 |                  |                                            |                                 |
| U 1258, 0925,84 | :20309 | WAIT.T20.0:      | JMP [WAIT.T20.0]                           | :LOOP HERE WAITING FOR          |
|                 | :20310 |                  |                                            | : CONSOLE TO RESPOND            |
|                 | :20311 |                  |                                            |                                 |
| U 1259, 65A0,15 | :20312 |                  | CLR LS[PREVIOUS.ERROR]                     | :CLEAR PREVIOUS ERROR NUMBER    |
|                 | :20313 |                  |                                            | : IN LS                         |
|                 | :20314 |                  |                                            |                                 |
| U 125A, B64A,15 | :20315 |                  | MOV LS[IDC] TO WR[0]                       | :STORE MODULE CODE IN LS TO     |
| U 125B, 3E8C,15 | :20316 |                  | MOV WR[0] TO LS[MODULE.NUM]                | : INDICATE IDC MODULE           |
|                 | :20317 |                  |                                            |                                 |
|                 | :20318 |                  |                                            |                                 |
| U 125C, B640,15 | :20319 |                  | MOV LS[#1] TO WR[0]                        | :SET WR0 = 1                    |
| U 125D, BE82,15 | :20320 |                  | MOV WR[0] TO LS[ERROR.NUMBER]              | :SET UP ERROR NUMBER = 1 IN LS  |
|                 | :20321 |                  |                                            |                                 |
| U 125E, E58A,15 | :20322 |                  | CLR LS[ERROR.MASK]                         | :SET UP ERROR MASK TO CHECK     |
|                 | :20323 |                  |                                            | : ALL BITS                      |
|                 | :20324 |                  |                                            |                                 |
| U 125F, 2F85,15 | :20325 |                  | CLR WR[2]                                  | :USE AS A LOOP COUNTER          |
| U 1260, B699,95 | :20326 |                  | MOV LS[AAAAAAA] TO WR[3]                   | :GET THE DATA PAT               |
|                 | :20327 |                  |                                            |                                 |
|                 | :20328 | CHK.NXT.PAT.T20: |                                            |                                 |
|                 | :20329 | LOOP.T20.1:      |                                            |                                 |
| U 1261, 17D8,15 | :20330 |                  | PORT.CONTROL [SEL.FIFO.A]                  | :SELECT FIFO A FROM THE PORT    |
| U 1262, 17C0,15 | :20331 |                  | PORT.CONTROL [CLEAR.FIFO.CNTR]             | :SET ADDRESS = 0                |
|                 | :20332 |                  |                                            |                                 |
| U 1263, A006,15 | :20333 |                  | MOV WR[3] TO WR[0]                         | :MOV THE PAT TO WR[0] FOR WRITE |
| U 1264, 17AC,15 | :20334 |                  | PORT.WRITE PORT.ADRS[DATA.WORD] WITH WR[0] | :WR LONG WORD OF DATA           |
|                 | :20335 |                  |                                            |                                 |
| U 1265, 17C0,15 | :20336 |                  | PORT.CONTROL [CLEAR.FIFO.CNTR]             | :SET FIFO ADDRESS = 0           |
|                 | :20337 |                  |                                            |                                 |
|                 | :20338 | READ.FIFO.T20.1: |                                            |                                 |
| U 1266, 9090,15 | :20339 |                  | MISC [SET.PORT.I.0]                        | :SELECT THE IDC                 |

U 1  
U 1  
  
U 1  
U 1  
  
U 1

ZZ-ENKCF-1.4 : ENKCF.MIC Test 20 I/O BUS IN<sup>L 1</sup>15-MAR-83 Fiche 3 Frame L1 Sequence 423  
: ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40 Page 417  
: ENKCF.MIC Test 20 I/O BUS IN AND OUT LATCHES TEST (M8388 IDC MODULE)  
:20395  
:20396

ZZ-E  
:  
:

U 13  
U 13

U 13  
U 13

U 13  
U 13

U 13  
U 13

U 13  
U 13

U 13  
U 13

U 13

U 13

U 13  
U 13

U 13  
U 13

U 13

U 1  
U 1  
U 1

U 1  
U 1



[illegible]

[illegible]

:20507 : CSR 11 H = L  
:20508 : CSR 10 H = L  
:20509 : PORT INSTR H = H  
:20510 : CPU P2 H = H  
:20511 : When CPU CLOCK H clocks the PAL, PCLR FIFO CNTR H will assert. This  
:20512 : signal is an input to the FIFO ADRS CNTR PALS and will clear all of  
:20513 : the address lines.  
:20514 :  
:20515 : The data to be written (0) is put on WR[0].  
:20516 : The CPU will then issue a PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0].  
:20517 : This instruction will be decoded in the PORT RD/WR DECODE PAL.  
:20518 : The inputs to this PAL will be:  
:20519 : CSR 17 H = H  
:20520 : CSR 14 H = L  
:20521 : CSR 13 H = H  
:20522 : CSR 12 H = L  
:20523 : CSR 11 H = H  
:20524 : CSR 10 H = L  
:20525 : PORT INSTR H = H  
:20526 : CPU P2 H = H  
:20527 : This combination of inputs will assert BYTE L and WRITE DATA L, and  
:20528 : STROBE DATA H.  
:20529 : BYTE L and WRITE DATA L are inputs to the PORT USEQ A & B PALS. This  
:20530 : combination of inputs will assert PWRITE FIFO H and PENB INREG B0 L.  
:20531 : PENB INREG B0 L and STROBE DATA H will enable the data from the I/O Bus  
:20532 : onto BUS IN. PWRITE FIFO H is ANDED with PSEL FIFOB H= H and PORT  
:20533 : CLK L. This combination of signals will assert WRITE FIFOB L which will  
:20534 : write the data which is on BUS IN into FIFOB.  
:20535 :  
:20536 : FIFO A will again be selected and the address lines cleared.  
:20537 : To read a byte of data from the FIFO, the CPU must first issue a  
:20538 : MISC [SET.PORT.I.0] which will assert SEL IDC H.  
:20539 : The CPU then will do a PORT.READ PORT ADRS[DATA.BYTE]. This  
:20540 : instruction will be decoded by PORT INTERFACE REGISTER PAL and the  
:20541 : PORT RD/WR DECODE PAL. The inputs to the PORT RD/WR DECODE PAL will  
:20542 : be:  
:20543 : CSR 17 H = H  
:20544 : CSR 14 H = L  
:20545 : CSR 13 H = L  
:20546 : CSR 12 H = L  
:20547 : CSR 11 H = H  
:20548 : CSR 10 H = L  
:20549 : PORT INSTR H = H  
:20550 : CPU P2 H = H  
:20551 : These inputs will assert READ DATA L and BYTE L.  
:20552 : These two signals are inputs to the PORT USEQ A and B PALS. These PALS  
:20553 : do not single-step with the CPU since they are clocked by CPU CLOCK H  
:20554 : which is a free running clock.  
:20555 : BRANCH TEST L will be asserted. When PORT USEQ PALS are clocked after  
:20556 : READ DATA L and BYTE L assert, the outputs of PORT USEQ A will be:  
:20557 : PORT NAD 0 H = H  
:20558 : PORT NAD 1 H = H  
:20559 : PORT NAD 2 H = H  
:20560 : BRANCH TEST L = H  
:20561 : PENB FIFO H = H

:20562 : PINCR FIFO CNTR H = H  
 :20563 : PWRITE FIFO H = L  
 :20564 :  
 :20565 : The only output of PORT USEQ B that will assert is PLOAD OUTREG B0 L.  
 :20566 :  
 :20567 : PENB FIFO H is ANDED with PSEL FIFOA H = H and asserts  
 :20568 : ENB FIFOA L which enables address 0 of FIFOA onto BUS OUT. PLOAD OUTREG  
 :20569 : B0 L clocks this data into the I/O BUS OUT LATCHES.  
 :20570 :

:20571 : At the same time, the PORT INTERFACE REG PAL is decoding the same  
 :20572 : instruction. The inputs to this PAL will be:  
 :20573 :

:20574 : CSR 17 H = H  
 :20575 : CSR 14 H = L  
 :20576 : READ PORT L = H  
 :20577 : CSR 12 H = L  
 :20578 : CSR 11 H = H  
 :20579 : CSR 10 H = L  
 :20580 : PORT INSTR H = H  
 :20581 : CPU P2 H = H  
 :20582 :

:20582 : On the next CPU CLOCK H, the outputs will be:  
 :20583 :

:20584 : SEL IDC H = H  
 :20585 : R2 L = H  
 :20586 : R1 L = L  
 :20587 : R0 L = H  
 :20588 : PSEL FIFOA H = H  
 :20589 :

:20590 : The CPU will then issue a MOV PORT TO LS[] instruction. This will  
 :20591 : assert READ PORT L which is an input to the PORT INTERFACE REG PAL.  
 :20592 : At the next CPU CLOCK, output READ PORT H will assert.  
 :20593 : This signal is one of the enables to the decoder of R<2:0>.  
 :20594 : READ PORT H is also ANDED with SEL IDC H, which will be asserted,  
 :20595 : and SEL ACC L = H. This combination of inputs will assert READ IDC L  
 :20596 : which is the enable to the decoder of R<2:0>.  
 :20597 : The combination of R<2:0> inputs that were previously latched by the  
 :20598 : PORT INTERFACE REG PAL, will assert SEL BYTE L. This signal is ORED  
 :20599 : with SEL WORD L and will assert ENB OUTREG L. ENB OUTREG L is the  
 :20600 : output enable for the I/O BUS OUT LATCHES and outputs the data onto  
 :20601 : the I/O BUS.  
 :20602 : READ IDC L which was previously asserted, directs the Tranceivers to  
 :20603 : take the data from the I/O BUS and put it on the Y BUS. The CPU then  
 :20604 : moves the data to an LS location.  
 :20605 :

:20606 : The CPU will then read and check the LS location.  
 :20607 :

:20608 : --  
 :20609 : \*\*\*\*\*

U 1289, B65E,15 :20610 : T.21:  
 :20611 : MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT 15 IN WR[0] FOR  
 :20612 : ; CONTROL AND STATUS WORD  
 U 128A, 3E80,15 :20613 : MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND  
 :20614 : ; STATUS WORD - BIT 15  
 :20615 : ; INDICATES START OF TEST TO  
 :20616 : ; CONSOLE

D 2

ZZ-ENKCF-1.4 : ENKCF.MIC Test 21 FIFO A AND 15-MAR-83 Fiche 3 Frame D2 Sequence 428  
: ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40 Page 422  
: ENKCF.MIC Test 21 FIFO A AND FIFO B UNIQUENESS TEST (M8388 IDC MODULE)

```

U 128B, 10E0,15 :20617 MISC [SET.CP.ATTN] ;ISSUE CPU ATTN TO CONSOLE
:20618 WAIT.T21.0:
U 128C, 0928,C4 :20619 JMP [WAIT.T21.0] ;LOOP HERE WAITING FOR
:20620 ; CONSOLE TO RESPOND
:20621
U 128D, 65A0,15 :20622 CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
:20623 ; IN LS
:20624
U 128E, B64A,15 :20625 MOV LS[IDC] TO WR[0] ;STORE MODULE CODE IN LS TO
U 128F, 3E8C,15 :20626 MOV WR[0] TO LS[MODULE.NUM] ; INDICATE IDC MODULE
:20627
:20628
U 1290, B640,15 :20629 MOV LS[#1] TO WR[0] ;SET WRO = 1
U 1291, BE82,15 :20630 MOV WR[0] TO LS[ERROR.NUMBER] ;SET UP ERROR NUMBER = 1 IN LS
:20631
U 1292, B62C,15 :20632 MOV LS[#FFFFFF00] TO WR[0] ;SET UP THE ERROR MASK TO
U 1293, 3E8A,15 :20633 MOV WR[0] TO LS[ERROR.MASK] ; CHECK THE LO BYTE
:20634
:20635 LOOP.T21.1:
U 1294, 17D8,15 :20636 PORT.CONTROL [SEL.FIFO.A] ;SELECT FIFO A
U 1295, 17C0,15 :20637 PORT.CONTROL [CLEAR.FIFO.CNTR] ;SET ADDRESS = 0
U 1296, B626,15 :20638 MOV LS[#FF] TO WR[0] ;GET THE DATA TO WRITE
U 1297, 97A8,15 :20639 PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;WR BYTE OF ONES TO FIFO A
U 1298, 97DC,15 :20640 PORT.CONTROL [SEL.FIFO.B] ;SELECT FIFO B
U 1299, 17C0,15 :20641 PORT.CONTROL [CLEAR.FIFO.CNTR] ;SET ADDRESS = 0
U 129A, 2F80,15 :20642 CLR WR[0] ;GET THE DATA TO WRITE
U 129B, 97A8,15 :20643 PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;WR 0 TO FIFO B ADRS=0
U 129C, 17D8,15 :20644 PORT.CONTROL [SEL.FIFO.A] ;SELECT FIFO A
U 129D, 17C0,15 :20645 PORT.CONTROL [CLEAR.FIFO.CNTR] ;SET ADDRESS = 0
:20646
:20647 READ.FIFO.T21:
U 129E, 9090,15 :20648 MISC [SET.PORT.I.0] ;SELECT THE IDC
U 129F, 1788,15 :20649 PORT.READ PORT.ADRS[DATA.BYTE] ;SET UP TO READ FIFO A ADRS =0
U 12A0, 3B32,15 :20650 MOV PORT TO LS[PORT0] ;READ THE DATA FROM THE FIFO
:20651 ; AND SAVE IT IN LS
U 12A1, 1080,15 :20652 MISC [SET.ACC.I.0] ;RETURN SELECT TO ACCELERATOR
U 12A2, 3732,15 :20653 MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
U 12A3, 3626,95 :20654 MOV LS[#FF] TO WR[1] ;SET UP EXPECTED DATA
U 12A4, 0869,3C :20655 JSR [CHECK.RESULT] ;CHECK THE DATA
U 12A5, 0929,44 :20656 JMP [LOOP.T21.1] ;LOOP HERE IF ERR & LOE IS SET
:20657
U 12A6, 8875,3C :20658 JSR [CLEANUP] ;CLEANUP THE CONDITIONS SET BY
:20659 ; THIS TEST
:20660
:20661 END.T21: ;END TEST
:20662
:20663
:20664

```

:20665 .PAGE 'Test 22 FIFO A ADDRESSING TEST (M8388 IDC MODULE)'  
:20666 .++  
:20667  
:20668 . Test Description:  
:20669  
:20670 . This test is designed to insure that there are no addressing problems  
:20671 . in FIFO A. This will be accomplished by first writing a background of  
:20672 . all zeros to FIFO A. Address 0 of FIFO A will then be written with all  
:20673 . ones and all other addresses in FIFO A will be checked to insure that  
:20674 . they were not corrupted.  
:20675 . The address counter will then be cleared and all locations that were  
:20676 . previously written with ones will be read to insure that the data is  
:20677 . correct. Then the next location will be written with all ones, and all  
:20678 . higher addresses in FIFO A will be read to insure that they still  
:20679 . contain all zeros. This will be repeated until all of FIFO A has been  
:20680 . written as ones.  
:20681  
:20682 . The procedure described above will then be repeated with first writing  
:20683 . a background of ones and then writing zeros to each location.  
:20684  
:20685 . Assumptions:  
:20686  
:20687 . It is assumed that all previous tests have run successfully.  
:20688  
:20689 . Test Steps:  
:20690  
:20691 . 1. Set up error mask, error number and module number in LS  
:20692 . (for error printouts) and clear the previous error number in LS.  
:20693 . 2. Write all zeros to FIFO A.  
:20694 . 3. Clear the FIFO A Address Counter.  
:20695 . 4. Use WR[2] to track the last address written and use  
:20696 . WR[3] to track the last address read.  
:20697 . 5. Write all ones to the first location in FIFO A.  
:20698 . 6. Check the remaining locations of FIFO A for all zeros.  
:20699 . 7. Read previously written all ones locations and check.  
:20700 . 8. Write the next location to all ones.  
:20701 . 9. Check the remaining locations in FIFO A for all zeros.  
:20702 . 10. Repeat Steps 7 - 10 until FIFO A has been written to all ones.  
:20703 . 11. Repeat the above sequence with writing a background of ones and  
:20704 . then writing zeros and reading.  
:20705 . Errors:  
:20706  
:20707 . Printout:  
:20708 . EXPECTED RECEIVED OTHER DATA  
:20709 . FIFO DATA FIFO DATA LS[ADDRESS.DATA]  
:20710 . LO WORD=LAST ADRS  
:20711 . WRITTEN  
:20712 . HI WORD=LAST ADRS  
:20713  
:20714  
:20715 . ERROR 1 - There is an addressing problem in FIFO A with a background of  
:20716 . zeros and writing ones.  
:20717  
:20718 . ERROR 2 - There is an addressing problem in FIFO A with a background of  
:20719 . ones and writing zeros.

U 1  
U 1  
U 1  
U 1  
U 1  
U 1

```

:20720 :
:20721 : Debug:
:20722 :
:20723 : Because of hardware restrictions, this test contains a special error
:20724 : loop. If loop is set, this test will go into a loop that is set up to
:20725 : increment the address lines of the FIFO. This is done to allow one to
:20726 : examine each of the address lines while looping to insure that each is
:20727 : incrementing correctly. The address lines should increment once for
:20728 : each byte write from the PORT.
:20729 : If the error loop is exited, and then a continue is issued, this test
:20730 : will cleanup and then end.
:20731 :
:20732 : ERROR 1:
:20733 : The CPU will issue a PORT.CONTROL[SEL.FIFO.A] instruction to select
:20734 : FIFO A from the PORT. The PORT.CONTROL[CLEAR.FIFO.CNTR] instruction
:20735 : will clear the address of FIFO A. Then each location of FIFO A will be
:20736 : written with [00] to provide a background of all zeros by issuing byte
:20737 : writes from the CPU. FIFO A address will then be cleared. Address 0
:20738 : will be written with [FF]. The entire FIFO A will then be read to
:20739 : insure that the data in all locations is correct.
:20740 : Then the next address will be written with [FF] and the entire
:20741 : FIFO checked again. This will be repeated until all location of the
:20742 : FIFO A have been written = [FF]. The address lines will increment by
:20743 : 1 for each byte write.
:20744 :
:20745 : ERROR 2:
:20746 : The same as ERROR 1 except a background of ones is used and then
:20747 : each location is written with [00].
:20748 : --
:20749 : *****
:20750 : LS[T8] = BACKGROUND DATA
:20751 : LS[T9] = USED AS A COUNTER TO TRACK THE FIFO ADDRESS
:20752 : LS[T10] = WRITE DATA
:20753 : LS[T11] = EXPECTED DATA
:20754 : WR[2] = LAST ADDRESS WRITTEN
:20755 : WR[3] = LAST ADDRESS READ IN UPPER WORD
:20756 : *****
:20757 : T.22:
:20758 : MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT 15 IN WR[0] FOR
:20759 : MOV WR[0] TO LS[CONTROL.STATUS] ; CONTROL AND STATUS WORD
:20760 : ;SET BIT 15 IN CONTROL AND
:20761 : ; STATUS WORD - BIT 15
:20762 : ; INDICATES START OF TEST TO
:20763 : ; CONSOLE
:20764 : MISC [SET.CP.ATTN] ;ISSUE CPU ATTN TO CONSOLE
:20765 : WAIT.T22.0:
:20766 : JMP [WAIT.T22.0] ;LOOP HERE WAITING FOR
:20767 : ; CONSOLE TO RESPOND
:20768 :
:20769 : CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
:20770 : ; IN LS
:20771 :
:20772 : MOV LS[IDC] TO WR[0] ;STORE MODULE CODE IN LS TO
:20773 : MOV WR[0] TO LS[MODULE.NUM] ; INDICATE IDC MODULE
:20774 :

```

U 12A7, B65E,15

U 12A8, 3E80,15

U 12A9, 10E0,15

U 12AA, 892A,A4

U 12AB, 65A0,15

U 12AC, B64A,15

U 12AD, 3E8C,15

ZZ-E

:

U 13

U 13

U 13

U 13

U 13

U 13

U 13

U 13

U 13

U 13

U 13

U 13

U 13

U 13

U 13

U 13

U 13

U 13

U 13

U 13

U 13

|   |   |
|---|---|
| U | 1 |
| U | 1 |
| U | 1 |
| U | 1 |
| U | 1 |
| U | 1 |



[illegible]

ZZ-ENKCF-1.4 :  
: ENKCF.MCR  
: ENKCF.MIC

ENKCF.MIC

Test 22 FIFO A ADD15-MAR-83

Fiche 3 Frame 12

Sequence 433

Page 427

ZZ-E

MICRO2 1M(01)

15-MAR-83 11:46:22

ENKCF Micro-diagnostic for IDC module

Rev 01.40

Test 22 FIFO A ADDRESSING TEST (M8388 IDC MODULE)

```
:20885
U 12E5, C061,95 :20886 ADD LS[#10000] TO WR[3] ;INCR THE LAST ADRS READ
U 12E6, FF12,15 :20887 INC LS[T9] ;INCR THE LAST ADRS READ
:20888
:20889
U 12E7, 4841,15 :20890 ADD WR[2] PLUS LS[#1] TO Q ;ADD 1 TO THE LAST ADRS WRITTEN
:20891 ; AND SAVE IT IN Q
:20892 CMP LS[T9] WITH Q, ;IS THIS THE NEXT ADRS TO BE
U 12E8, DA12,35 :20893 DT(LONG)&SET.ALU.CC ; WRITTEN?
U 12E9, 092E,C1 :20894 JMP.IF[NEQ] TO [CHK.READ.T22] ;JMP IF NOT
:20895
U 12EA, 2045,15 :20896 INC WR[2] ;INC THE LAST ADRS WRITTEN
U 12EB, 892C,C4 :20897 JMP [WRITE.FIFO.A.T22] ;GO WRITE THE NEXT ADDRESS
:20898
:20899 CHK.READ.T22:
U 12EC, B652,15 :20900 MOV LS[#200] TO WR[0] ;LAST LOCATION IN FIFO READ?
:20901 CMP LS[T9] WITH WR[0], ;NEXT LOCATION = 0?
U 12ED, CE12,35 :20902 DT(LONG)&SET.ALU.CC
U 12EE, 092D,C1 :20903 JMP.IF[NEQ] TO [READ.FIFO.A.T22] ;JMP IF NOT 0
:20904
U 12EF, 3614,15 :20905 MOV LS[T10] TO WR[0] ;CHANGE THE EXPECTED DATA
U 12F0, 3E16,15 :20906 MOV WR[0] TO LS[T11] ; TO THE DATA BEING WRITTEN
U 12F1, 2F87,95 :20907 CLR WR[3] ;LAST READ ADDRESS = 0
U 12F2, 6512,15 :20908 CLR LS[T9] ;LAST READ ADDRESS = 0
U 12F3, 17C0,15 :20909 PORT.CONTROL [CLEAR.FIFO.CNTR] ;CLR THE FIFO ADRS CNTR
U 12F4, 092D,C4 :20910 JMP [READ.FIFO.A.T22] ;GO READ FOR ONES
:20911
U 12F5, 5B00,14 :20912 RETURN.T22:
:20913 RETURN
:20914
:20915 ;-----
:20916 ;-----
:20917 ;ERROR LOOP
:20918
:20919 LOOP.T22.1:
U 12F6, 2F80,15 :20920 CLR WR[0] ;DATA TO WRITE
U 12F7, 97A8,15 :20921 PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;WRITE TO INCR THE ADRS
:20922
:20923 ;EXPECTED AND RECEIVED DATA
:20924 ; IS MEANINGLESS WHEN IN THE
:20925 ; ERROR LOOP
U 12F8, 0869,3C :20926 JSR [CHECK.RESULT]
U 12F9, 892F,64 :20927 JMP [LOOP.T22.1] ;LOOP HERE IF ERR & LOE IS SET
:20928 ;IF STOP LOOP AND CONTINUE,
U 12FA, 092F,B4 :20929 JMP [CLEANUP.T22] ; CLEANUP AND END TEST
:20930
:20931 ;-----
:20932
:20933 CLEANUP.T22:
U 12FB, 8875,3C :20934 JSR [CLEANUP]
:20935
:20936 END.T22:
:20937
:20938
:20939 ;CLEANUP THE CONDITIONS SET BY
:20940 ; THIS TEST
:20941 ;END TEST
```

20940  
20941  
20942  
20943  
20944  
20945  
20946  
20947  
20948  
20949  
20950  
20951  
20952  
20953  
20954  
20955  
20956  
20957  
20958  
20959  
20960  
20961  
20962  
20963  
20964  
20965  
20966  
20967  
20968  
20969  
20970  
20971  
20972  
20973  
20974  
20975  
20976  
20977  
20978  
20979  
20980  
20981  
20982  
20983  
20984  
20985  
20986  
20987  
20988  
20989  
20990  
20991  
20992  
20993  
20994

PAGE "Test 23 FIFO B ADDRESSING TEST (M8388 IDC MODULE)"  
++  
Test Description:  
This test is designed to insure that there are no addressing problems in FIFO B. This will be accomplished by first writing a background of all zeros to FIFO B. Address 0 of FIFO B will then be written with all ones and all other addresses in FIFO B will be checked to insure that they were not corrupted.  
The address counter will then be cleared and all locations that were previously written with ones will be read to insure that the data is correct. Then the next location will be written with all ones, and all higher addresses in FIFO B will be read to insure that they still contain all zeros. This will be repeated until all of FIFO B has been written as ones.  
The procedure described above will then be repeated with first writing a background of ones and then writing zeros to each location.  
Assumptions:  
It is assumed that all previous tests have run successfully.  
Test Steps:  
1. Set up error mask, error number and module number in LS (for error printouts) and clear the previous error number in LS.  
2. Write all zeros to FIFO B.  
3. Clear the FIFO B Address Counter.  
4. Use WR[2] to track the last address written and use WR[3] to track the last address read.  
5. Write all ones to the first location in FIFO B.  
6. Check the remaining locations of FIFO B for all zeros.  
7. Read previously written all ones locations and check.  
8. Write the next location to all ones.  
9. Check the remaining locations in FIFO B for all zeros.  
10. Repeat Steps 7 - 10 until FIFO B has been written to all ones.  
11. Repeat the above sequence with first writing a background of ones and then writing zeros and reading.  
Errors:  
Printout:  

EXPECTED  
FIFO DATA

RECEIVED  
FIFO DATA

OTHER DATA  
LS[ADDRESS.DATA]  
LO WORD=LAST ADRS  
WRITTEN  
HI WORD=LAST ADRS  
READ

ERROR 1 - There is an addressing problem in FIFO B with a background of

U 13  
U 13  
U 13  
  
U 13  
U 13  
U 13  
  
U 13  
U 13  
U 13  
  
U 13  
U 13  
U 13

```

:20995 : zeros and writing ones.
:20996 :
:20997 : ERROR 2 - There is an addressing problem in FIFO B with a background of
:20998 : ones and writing zeros.
:20999 :
:21000 : Debug:
:21001 :
:21002 : Because of hardware restrictions, this test conatins a special error
:21003 : loop. If loop is set, this test will go into a loop that is set up to
:21004 : increment the address lines of the FIFO. This is done to allow one to
:21005 : examine each of the address lines while looping to insure that each is
:21006 : incrementing correctly. The address lines should increment once for
:21007 : each byte write from the PORT.
:21008 : If the error loop is exited, and then a continue is issued, this test
:21009 : will cleanup and then end.
:21010 :
:21011 : ERROR 1:
:21012 : The CPU will issue a PORT.CONTROL[SEL.FIFO.B] instruction to select
:21013 : FIFO B from the PORT. The PORT.CONTROL[CLEAR.FIFO.CNTR] instruction
:21014 : will clear the address of FIFO B. Then each location of FIFO B will be
:21015 : written with [00] to provide a background of all zeros by issuing byte
:21016 : writes from the CPU. FIFO B address will then be cleared. Address 0
:21017 : will be written with [FF]. The entire FIFO B will then be read to
:21018 : insure that the data in all locations is correct.
:21019 : Then the next address will be written with [FF] and the entire
:21020 : FIFO checked again. This will be repeated until all location of the
:21021 : FIFO B have been written = [FF]. The address lines will increment by
:21022 : 1 for each byte write.
:21023 :
:21024 : ERROR 2:
:21025 : The same as ERROR 1 except a background of ones is used and then
:21026 : each location is written with [00].
:21027 : --
:21028 : *****
:21029 : LS[T8] = BACKGROUND DATA
:21030 : LS[T9] = USED AS A COUNTER TO TRACK THE FIFO ADDRESS
:21031 : LS[T10] = WRITE DATA
:21032 : LS[T11] = EXPECTED DATA
:21033 : WR[2] = LAST ADDRESS WRITTEN
:21034 : WR[3] = LAST ADDRESS READ IN UPPER WORD
:21035 : *****
:21036 : T.23:
:21037 : MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT 15 IN WR[0] FOR
:21038 : ; CONTROL AND STATUS WORD
:21039 : MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND
:21040 : ; STATUS WORD - BIT 15
:21041 : ; INDICATES START OF TEST TO
:21042 : ; CONSOLE
:21043 : MISC [SET.CP.ATTN] ;ISSUE CPU ATTN TO CONSOLE
:21044 : WAIT.T23.0:
:21045 : JMP [WAIT.T23.0] ;LOOP HERE WAITING FOR
:21046 : ; CONSOLE TO RESPOND
:21047 :
:21048 : CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
:21049 : ; IN LS

```

013

```

U 1319, 2045,15 :21105
 :21106 INC WR[2] ;INCR THE ADDRESS COUNTER
 :21107 CMP WR[2] WITH LS[#200], ;FIFO B WRITTEN WITH ALL ZEROS?
U 131A, 4F53,35 :21108 DT(LONG)&SET.ALU.CC
U 131B, 0931,71 :21109 JMP.IF[NEQ] TO [WRITE.FIFO.ALL.T23]
U 131C, 5B00,14 :21110 RETURN
 :21111
 :21112 ;-----
 :21113
 :21114 ;THIS IS A SUBROUTINE THAT WILL WRITE ONE LOCATION AT A TIME WITH THE DATA
 :21115 ;THAT IS IN LS[T10]. THE REST OF THE FIFO WILL THEN BE READ AND CHECKED.
 :21116 ;THE BACKGROUND DATA MUST BE IN LS[T8] WHEN THIS ROUTINE IS CALLED.
 :21117
U 131D, 2F85,15 :21118 TEST.FIFO.T23:
U 131E, 2F87,95 :21119 CLR WR[2] ;ADRS CNTR= LAST ADRS WRITTEN
 :21120 CLR WR[3] ;ADRS CNTR= LAST ADRS READ
 :21121 ; WILL BE IN THE HI WORD
 :21122 ; FOR PRINTOUT
U 131F, 6512,15 :21123 CLR LS[T9] ;ADRS CNTR = LAST ADRS READ
 :21124 ; WILL BE IN LOW BYTE
 :21125
U 1320, 17C0,15 :21126 PORT.CONTROL [CLEAR.FIFO.CNTR] ;SET FIFO B ADRS = 0
 :21127
 :21128 WRITE.FIFO.B.T23:
U 1321, 4F53,35 :21129 CMP WR[2] WITH LS[#200], ;DONE?
U 1322, 0934,A9 :21130 DT(LONG)&SET.ALU.CC
 :21131 JMP.IF[EQL] TO [RETURN.T23] ;JMP IF DONE
 :21132
U 1323, 3614,15 :21133 MOV LS[T10] TO WR[0] ;GET THE DATA
U 1324, 97A8,15 :21134 PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;WR FIFO B
 :21135
U 1325, 3610,95 :21136 MOV LS[T8] TO WR[1] ;SET UP THE EXPECTED DATA
U 1326, BE16,95 :21137 MOV WR[1] TO LS[T11] ; SAVE IT IS LS[T11]
 :21138
U 1327, C061,95 :21139 ADD LS[#10000] TO WR[3] ;INCR THE LAST ADRS READ
U 1328, FF12,15 :21140 INC LS[T9] ;INCR THE LAST ADRS READ
 :21141
U 1329, B652,15 :21142 MOV LS[#200] TO WR[0] ;NEXT READ AT ZERO?
 :21143 CMP LS[T9] WITH WR[0],
U 132A, CE12,35 :21144 DT(LONG)&SET.ALU.CC
U 132B, 8933,11 :21145 JMP.IF[NEQ] TO [READ.FIFO.B.T23] ;JMP IF NOT 0
 :21146
U 132C, 3614,15 :21147 MOV LS[T10] TO WR[0] ;CHANGE THE EXPECTED DATA
U 132D, 3E16,15 :21148 MOV WR[0] TO LS[T11] ; TO DATA WRITTEN
 :21149
U 132E, 6512,15 :21150 CLR LS[T9] ;CLR THE ADDRESS TRACKER
U 132F, 2F87,95 :21151 CLR WR[3] ;CLR THE PRINTOUT COUNTER
U 1330, 17C0,15 :21152 PORT.CONTROL [CLEAR.FIFO.CNTR] ;CLR THE FIFO ADRS CNTR
 :21153
 :21154 READ.FIFO.B.T23:
U 1331, 9090,15 :21155 MISC [SET.PORT.I.O] ;SELECT THE IDC
U 1332, 1788,15 :21156 PORT.READ PORT.ADRS[DATA.BYTE] ;SET UP TO READ THE FIFO
U 1333, 3B32,15 :21157 MOV PORT TO LS[PORT0] ;READ THE DATA FROM FIFO AND
 :21158 ; SAVE IT IN LS
U 1334, 3E89,15 :21159 MOV WR[2] TO LS[ADDRESS.DATA] ;SET UP THE ADDRESS INFORMATION

```

U 13  
 U 13

U 13  
 U 13

U 13  
 U 13

U 13

U 13  
 U 13

U 13  
 U 13

```

U 1335, ED89,95 :21160 BIS WR[3] TO LS[ADDRESS.DATA] ; FOR PRINTOUT
U 1336, 3732,15 :21161 MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
U 1337, 3616,95 :21162 MOV LS[T11] TO WR[1] ;SET UP EXPECTED DATA
U 1338, 0869,3C :21163 JSR [CHECK.RESULT] ;CHECK THE DATA
U 1339, 0934,84 :21164 JMP [LOOP.T23.1] ;LOOP HERE IF ERR & LOE IS SET
 :21165
U 133A, C061,95 :21166 ADD LS[#10000] TO WR[3] ;INCR THE LAST ADRS READ
U 133B, FF12,15 :21167 INC LS[T9] ;INCR THE LAST ADRS READ
 :21168
U 133C, 4841,15 :21169 ADD WR[2] PLUS LS[#1] TO Q ;ADD 1 TO THE LAST ADRS WRITTEN
 :21170
 :21171 CMP LS[T9] WITH Q, ; AND SAVE IT IN Q
U 133D, DA12,35 :21172 DT(LONG)&SET.ALU.CC ;IS THIS THE NEXT ADRS TO BE
U 133E, 0934,11 :21173 JMP.IF[NEQ] TO [CHK.READ.T23] ; WRITTEN?
 :21174
 :21175 INC WR[2] ;INC THE LAST ADRS WRITTEN
U 133F, 2045,15 :21176 JMP [WRITE.FIFO.B.T23] ;GO WRITE THE NEXT ADDRESS
U 1340, 0932,14 :21177
 :21178
U 1341, B652,15 :21179 CHK.READ.T23: ;LAST LOCATION IN FIFO READ?
 :21180 MOV LS[#200] TO WR[0] ;NEXT LOCATION = 0?
 :21181 CMP LS[T9] WITH WR[0],
U 1342, CE12,35 :21182 DT(LONG)&SET.ALU.CC
U 1343, 8933,11 :21183 JMP.IF[NEQ] TO [READ.FIFO.B.T23] ;JMP IF NOT 0
 :21184
U 1344, 3614,15 :21185 MOV LS[T10] TO WR[0] ;CHANGE THE EXPECTED DATA
U 1345, 3E16,15 :21186 MOV WR[0] TO LS[T11] ; TO DATA WRITTEN
U 1346, 2F87,95 :21187 CLR WR[3] ;LAST READ ADDRESS = 0
U 1347, 6512,15 :21188 CLR LS[T9] ;LAST READ ADDRESS = 0
U 1348, 17C0,15 :21189 PORT.CONTROL [CLEAR.FIFO.CNTR] ;CLR THE FIFO ADRS CNTR
U 1349, 8933,14 :21190 JMP [READ.FIFO.B.T23] ;GO READ FOR ONES
 :21191
U 134A, 5B00,14 :21192 RETURN.T23:
 :21193 RETURN
 :21194
 :21195 -----
 :21196 ;ERROR LOOP
 :21197
U 134B, 2F80,15 :21198 LOOP.T23.1:
U 134C, 97A8,15 :21199 CLR WR[0] ;DATA TO WRITE
 :21200 PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;WRITE TO INCR THE ADRS
 :21201
 :21202 ;EXPECTED AND RECEIVED DATA
 :21203 ; IS MEANINGLESS WHILE IN THE
U 134D, 0869,3C :21204 JSR [CHECK.RESULT] ; THE ERROR LOOP
U 134E, 0934,84 :21205 JMP [LOOP.T23.1] ;LOOP HERE IF ERR & LOE IS SET
 :21206 ;IF STOP LOOP AND CONTINUE,
U 134F, 0935,04 :21207 JMP [CLEANUP.T23] ; CLEANUP AND END TEST
 :21208
 :21209 -----
U 1350, 8875,3C :21210 CLEANUP.T23:
 :21211 JSR [CLEANUP] ;CLEANUP THE CONDITIONS SET BY
 :21212
 :21213 END.T23: ; THIS TEST
 :21214 ;END TEST

```

1  
ZZ-ENKCF-1.4 ; ENKCF.MIC Test 23 FIFO B ADD15-MAR-83 B 3 Fiche 3 Frame B3 Sequence 439  
: ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40 Page 433  
: ENKCF.MIC Test 23 FIFO B ADDRESSING TEST (M8388 IDC MODULE)  
;21215

ZZ-1  
:  
:  
U 1  
U 1  
U 1  
U 1  
U 1  
U 1  
U 1  
U 1  
U 1  
U 1



```

:21216 PAGE 'Test 24 PARALLEL IN/OUT DATA SHIFT REGISTER TEST (M8388 IDC MODULE)'
:21217 :++
:21218 :
:21219 : Test Description:
:21220 :
:21221 : This test is designed to insure that the Data Shift Register can be
:21222 : parallel loaded and read correctly. The data pattern will be loaded
:21223 : into the Data Shift Register from FIFO A address 0 and then enabled
:21224 : into FIFO A address 1 for the CPU to read and check.
:21225 : The data patterns that will be used are:
:21226 : Data Pattern 1 = AA
:21227 : Data Pattern 2 = 55
:21228 : Data Pattern 3 = 33
:21229 : Data Pattern 4 = 0F
:21230 :
:21231 : After these data patterns have been checked, the UCLR DSR function will
:21232 : be tested to insure that it indeed does clear the DSR.
:21233 :
:21234 : Assumptions:
:21235 :
:21236 : It is assumed that all previous test have run successfully.
:21237 :
:21238 : Test Steps:
:21239 :
:21240 : 1. Set up error mask, error number and module number in LS
:21241 : (for error printouts) and clear the previous error number in LS.
:21242 : 2. Force the IDC microcode to DIAG.IDLE.
:21243 : 3. Clear the FIFO A Address Counter.
:21244 : 4. Clear FIFO A addresses 0 and 1.
:21245 : 5. Write the data pattern to FIFO A address 0.
:21246 : 6. Clear the FIFO A Address Counter.
:21247 : 7. Set up the CSR to branch to the correct IDC routine.
:21248 :
:21249 :
:21250 : *****
:21251 : * IDC *
:21252 : *****
:21253 :
:21254 : IDC1. Enable FIFO A onto BUS OUT.
:21255 : IDC2. Issue ULOAD DSR.
:21256 : IDC3. UENB DSR onto BUS IN.
:21257 : IDC4. Write the data to FIFO A address 1.
:21258 : IDC5. Issue a XFER.REQ to flag the CPU.
:21259 :
:21260 : 8. Wait for a XFER.REQ from IDC.
:21261 : 9. Clear CSR F<2:0> and send XFER GRANT to clear XFER.REQ.
:21262 : 10. Check the data in FIFO A address 1.
:21263 : 11. Get the next data pattern and execute Steps 3 - 10 until all data
:21264 : patterns have been checked.
:21265 : 12. Increment the error number.
:21266 : 13. Clear the FIFO A Address Counter.
:21267 : 14. Write FFFFFFFF to FIFO A.
:21268 : 15. Clear the FIFO A Address Counter.
:21269 : 16. Set up the CSR to branch to the correct IDC routine.
:21270 :

```

```

:21271 :
:21272 :
:21273 : *****
:21274 : * IDC *
:21275 : *****
:21276 :
:21277 : IDC6. Load the DSR with the data in FIFO A.
:21278 : IDC7. Issue a UCLR DSR.
:21279 : IDC8. Enable the DSR into FIFO A address=1.
:21280 : IDC9. Issue a XFER.REQ to flag the CPU.
:21281 :
:21282 : 17. Wait for a XFER.REQ from IDC.
:21283 : 18. Clear CSR F<2:0> and send XFER GRANT to clear XFER.REQ.
:21284 : 19. Check the data in FIFO A.
:21285 : 20. Clear IDC.
:21286 : 21. End test.
:21287 :
:21288 : Errors:
:21289 :
:21290 : Printout:
:21291 : EXPECTED RECEIVED
:21292 : DSR DATA DSR DATA
:21293 : FROM FIFO A
:21294 :
:21295 : ERROR 1 - The Data Shift Register or the DSR Buffer failed.
:21296 :
:21297 : ERROR 2 - The UCLR DSR function failed.
:21298 :
:21299 : Debug:
:21300 : ERROR 1:
:21301 : FIFO A address 0 will contain the data pattern to be loaded into the
:21302 : DSR and FIFO A address 1 will be cleared to be used to store the
:21303 : data read from the DSR. The IDC microcode will branch to an instruction
:21304 : that will select FIFO A from the IDC microcode. The next IDC
:21305 : instruction is:
:21306 : -----
:21307 : 1DE:
:21308 : TEST.DSR.1:
:21309 :
:21310 : ENB.FIFO/ON, ;ENABLE FIFO A ONTO BUS OUT
:21311 : LOAD.DSR/ON, ; AND LOAD THE DATA INTO THE DSR
:21312 : INCR.FIFO.CNTR/ON, ; INCREMENT THE FIFO A CNTR TO 1
:21313 : NAD/TEST.DSR.2
:21314 : -----
:21315 : UENB FIFO H will be asserted by this instruction. The signal is
:21316 : ANDED with USEL FIFO A H = H, the output of which is ENB FIFOA L = L.
:21317 : ENB FIFOA L will enable the data from FIFO A address 0 onto
:21318 : BUS OUT <7:0>. This is the input BUS for the DSR. ULOAD DSR L is also
:21319 : asserted by this instruction. This signal is inverted to LOAD DSR H = H
:21320 : which will load the data from BUS OUT <7:0> into the DSR.
:21321 : Also asserted by this instruction is UINCR FIFO CNTR H which is part
:21322 : of the clock signal to FIFO A. Since USEL FIFO A H is also asserted,
:21323 : at the next CURRENT CLK L, the FIFO A address will be incremented
:21324 : to 1.
:21325 :

```

[illegible]

```

U 135A, B62C,15 :21381 MOV LS[FFFFFF00] TO WR[0] ;SET UP THE ERROR MASK TO CHECK
U 135B, 3E8A,15 :21382 MOV WR[0] TO LS[ERROR.MASK] ; THE LOW BYTE
:21383
U 135C, 8872,EC :21384 JSR [DIAG.CODE] ;FORCE THE IDC TO DIAG.IDLE
:21385
U 135D, 2F85,15 :21386 CLR WR[2] ;USE AS A LOOP CNTR
U 135E, B699,95 :21387 MOV LS[AAAAAAAA] TO WR[3] ;GET THE DATA PATTERN
:21388
:21389 CHK.NXT.PAT.T24:
:21390 LOOP.T24.1:
:21391
:21392
U 135F, 17D8,15 :21393 PORT.CONTROL [SEL.FIFO.A]
U 1360, 17C0,15 :21394 PORT.CONTROL [CLEAR.FIFO.CNTR] ;CLR THE FIFO A ADRS CNTR
:21395
U 1361, 2F80,15 :21396 CLR WR[0]
U 1362, 17AC,15 :21397 PORT.WRITE PORT.ADRS[DATA.WORD] WITH WR[0] ;CLR ADRS 0 AND 1 OF FIFO A
:21398
:21399
U 1363, 17D8,15 :21400 PORT.CONTROL [SEL.FIFO.A]
U 1364, 17C0,15 :21401 PORT.CONTROL [CLEAR.FIFO.CNTR] ;CLR THE FIFO A ADRS CNTR
:21402
U 1365, A006,15 :21403 MOV WR[3] TO WR[0] ;MOV PAT TO WR[0] FOR WRITE
U 1366, 97A8,15 :21404 PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;WRITE FIFO A ADRS 0 = PAT
:21405
U 1367, 17C0,15 :21406 PORT.CONTROL [CLEAR.FIFO.CNTR] ;CLR THE FIFO A ADRS CNTR
U 1368, 3720,15 :21407 MOV LS[SETCSR.MAINT] TO WR[0] ;SET UP THE CSR TO BRANCH TO
U 1369, 3712,95 :21408 MOV LS[SETCSR.CRDY] TO WR[1] ; THE CORRECT IDC ROUTINE
U 136A, AEC2,15 :21409 BIS WR[1] TO WR[0]
U 136B, 370A,95 :21410 MOV LS[SETCSR.F5] TO WR[1]
U 136C, AEC2,15 :21411 BIS WR[1] TO WR[0]
U 136D, 17A0,15 :21412 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:21413 ;MAINT=1,CRDY=1,F<2:0>=101
:21414
:21415
:21416 *****
:21417 * IDC *
:21418 *****
:21419
:21420 :018:
:21421 :=000
:21422 :DIAG.IDLE:
:21423
:21424 BEN0/F0,
:21425 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:21426 BEN2/F2,
:21427 NAD/DIAG.IDLE
:21428
:21429 :01D:
:21430 :=101
:21431
:21432 BEN0/CRDY.L,
:21433 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:21434 NAD/TEST.FUNC5 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:21435

```

```

:21436 :074:
:21437 :=1*0
:21438 :TEST.DSR.0:
:21439 :
:21440 : UCMD/SET.FIFOA, ;SELECT FIFO A
:21441 : NAD/TEST.DSR.1
:21442 :-----
:21443 :1DE:
:21444 :TEST.DSR.1:
:21445 :
:21446 : ENB.FIFO/ON, ;ENABLE FIFO A ONTO BUS OUT
:21447 : LOAD.DSR/ON, ; AND LOAD THE DATA INTO THE DSR
:21448 : INCR.FIFO.CNTR/ON, ; INCREMENT THE FIFO A CNTR TO 1
:21449 : NAD/TEST.DSR.2
:21450 :-----
:21451 :1DF:
:21452 :TEST.DSR.2:
:21453 :
:21454 : ENB.DSR/ON, ;LOAD THE DSR ONTO BUS IN
:21455 : WRT.FIFO/ON, ;WRITE THE DATA INTO FIFO A ADRS=1
:21456 : NAD/XFER.REQ ;SEND XFER REQ TO FLAG THE CPU
:21457 :-----
:21458 :1D8:
:21459 :XFER.REQ:
:21460 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:21461 : NAD/DIAG.WAIT
:21462 :-----
:21463 :1D7:
:21464 :DIAG.WAIT:
:21465 :
:21466 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:21467 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:21468 :-----
:21469 :159:
:21470 :=*0*
:21471 :DIAG.WAIT1:
:21472 :
:21473 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:21474 : ; COMMAND FROM THE CPU
:21475 :-----
:21476 :15B:
:21477 :=*1*
:21478 :DIAG.WAIT2:
:21479 :
:21480 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:21481 : ; WAIT SOME MORE
:21482 :-----
:21483 :WAIT.IDC.T24.1:
:21484 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ TO SET
:21485 : JMP [IDC.DONE.T24.1] ;XFER REQ ASSERTED
:21486 : JMP [WAIT.IDC.T24.1] ;WAIT SOME MORE
:21487 :
:21488 :IDC.DONE.T24.1:
:21489 : MOV LS[SETCSR.F0] TO WR[0] ;SET UP TO CLR CSR F<2:0>
:21490 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ; WRITE THE CSR

```

U 136E, DB00,1A  
 U 136F, 0937,14  
 U 1370, 8936,E4

U 1371, B700,15  
 U 1372, 17A0,15

|              |   |           |                                                                     |                           |                                                           |                           |          |                |
|--------------|---|-----------|---------------------------------------------------------------------|---------------------------|-----------------------------------------------------------|---------------------------|----------|----------------|
| ZZ-ENKCF-1.4 | : | ENKCF.MIC | Test 24 PARALLEL IN/OUT DATA SHIFT REGISTER TEST (M8388 IDC MODULE) | H 3<br>15-MAR-83 11:46:22 | Fiche 3 Frame H3<br>ENKCF Micro-diagnostic for IDC module | Sequence 445<br>Rev 01.40 | Page 439 | ZZ-E<br>:<br>: |
| :            | : | ENKCF.MCR |                                                                     |                           |                                                           |                           |          |                |
| :            | : | ENKCF.MIC |                                                                     |                           |                                                           |                           |          |                |

  

|                 |   |       |                                            |   |                               |
|-----------------|---|-------|--------------------------------------------|---|-------------------------------|
| U 1373, 90FA,15 | : | 21491 | MISC2 [TRANSFER.GRANT]                     | : | SEND GRANT TO CLR XFER REQ    |
|                 | : | 21492 |                                            |   |                               |
|                 | : | 21493 | READ.FIFO.T24.1:                           |   |                               |
| U 1374, 9090,15 | : | 21494 | MISC [SET.PORT.I.0]                        | : | SELECT THE IDC                |
| U 1375, 1788,15 | : | 21495 | PORT.READ PORT.ADRS[DATA.BYTE]             | : | SET UP TO READ FIFO A ADRS=1  |
| U 1376, 3B32,15 | : | 21496 | MOV PORT TO LS[PORT0]                      | : | READ THE DATA FROM FIFO AND   |
|                 | : | 21497 |                                            | : | SAVE IT IN LS                 |
| U 1377, 3732,15 | : | 21498 | MOV LS[PORT0] TO WR[0]                     | : | SET UP RECEIVED DATA          |
| U 1378, 2006,95 | : | 21499 | MOV WR[3] TO WR[1]                         | : | SET UP THE EXPECTED DATA      |
|                 | : | 21500 |                                            |   |                               |
| U 1379, 0869,3C | : | 21501 | JSR [CHECK.RESULT]                         | : | CHECK THE DATA                |
| U 137A, 0935,F4 | : | 21502 | JMP [LOOP.T24.1]                           | : | LOOP HERE IF ERR & LOE IS SET |
|                 | : | 21503 |                                            |   |                               |
|                 | : | 21504 |                                            |   |                               |
|                 | : | 21505 | CMP WR[2] WITH LS[ZERO],                   | : | FIRST LOOP COUNT?             |
| U 137B, CF9D,35 | : | 21506 | DT(LONG)&SET.ALU.CC                        |   |                               |
| U 137C, 8938,01 | : | 21507 | JMP.IF[NEQ] TO [CHK.LPCNT.T24.1]           | : | JMP IF NOT FIRST              |
| U 137D, 369B,95 | : | 21508 | MOV LS[#55555555] TO WR[3]                 | : | GET DATA PATTERN 2            |
| U 137E, 2045,15 | : | 21509 | INC WR[2]                                  | : | INCR THE LOOP COUNT           |
| U 137F, 0935,F4 | : | 21510 | JMP [CHK.NXT.PAT.T24]                      | : | CHECK THE NEXT PATTERN        |
|                 | : | 21511 |                                            |   |                               |
|                 | : | 21512 | CHK.LPCNT.T24.1:                           |   |                               |
|                 | : | 21513 | CMP WR[2] WITH LS[#1],                     | : | SECOND LOOP COUNT?            |
| U 1380, 4F41,35 | : | 21514 | DT(LONG)&SET.ALU.CC                        |   |                               |
| U 1381, 8938,51 | : | 21515 | JMP.IF[NEQ] TO [CHK.LPCNT.T24.2]           | : | JMP IF NOT                    |
| U 1382, B6C5,95 | : | 21516 | MOV LS[#33333333] TO WR[3]                 | : | GET DATA PATTERN 3            |
| U 1383, 2045,15 | : | 21517 | INC WR[2]                                  | : | INCR THE LOOP COUNT           |
| U 1384, 0935,F4 | : | 21518 | JMP [CHK.NXT.PAT.T24]                      | : | CHECK THE NEXT PATTERN        |
|                 | : | 21519 |                                            |   |                               |
|                 | : | 21520 | CHK.LPCNT.T24.2:                           |   |                               |
|                 | : | 21521 | CMP WR[2] WITH LS[#2],                     | : | THIRD LOOP COUNT?             |
| U 1385, CF43,35 | : | 21522 | DT(LONG)&SET.ALU.CC                        |   |                               |
| U 1386, 8938,A1 | : | 21523 | JMP.IF[NEQ] TO [TEST.CLR.T24]              | : | JMP IF DONE                   |
| U 1387, 36C7,95 | : | 21524 | MOV LS[#0F0F0F0F] TO WR[3]                 | : | GET DATA PATTERN 4            |
| U 1388, 2045,15 | : | 21525 | INC WR[2]                                  | : | INCR THE LOOP COUNT           |
| U 1389, 0935,F4 | : | 21526 | JMP [CHK.NXT.PAT.T24]                      | : | CHECK THE NEXT PATTERN        |
|                 | : | 21527 |                                            |   |                               |
|                 | : | 21528 | TEST.CLR.T24:                              |   |                               |
| U 138A, FF82,15 | : | 21529 | INC LS[ERROR.NUMBER]                       | : | ERROR 2                       |
|                 | : | 21530 |                                            |   |                               |
|                 | : | 21531 | LOOP.T24.2:                                |   |                               |
|                 | : | 21532 |                                            |   |                               |
|                 | : | 21533 |                                            |   |                               |
| U 138B, 17D8,15 | : | 21534 | PORT.CONTROL [SEL.FIFO.A]                  |   |                               |
| U 138C, 17C0,15 | : | 21535 | PORT.CONTROL [CLEAR.FIFO.CNTR]             | : | CLR THE FIFO A ADRS CNTR      |
|                 | : | 21536 |                                            |   |                               |
| U 138D, B69E,15 | : | 21537 | MOV LS[FFFFFFFF] TO WR[0]                  | : | WRITE ONES TO FIFO A          |
| U 138E, 17AC,15 | : | 21538 | PORT.WRITE PORT.ADRS[DATA.WORD] WITH WR[0] | : | ADDRESS ZERO AND 1            |
| U 138F, 17C0,15 | : | 21539 | PORT.CONTROL [CLEAR.FIFO.CNTR]             | : | CLR FIFO A ADRS CNTR          |
|                 | : | 21540 |                                            |   |                               |
| U 1390, 3712,95 | : | 21541 | MOV LS[SETCSR.CRDY] TO WR[1]               | : | SET UP THE CSR TO BRANCH TO   |
| U 1391, B70C,15 | : | 21542 | MOV LS[SETCSR.F6] TO WR[0]                 | : | THE CORRECT IDC ROUTINE       |
| U 1392, AEC2,15 | : | 21543 | BIS WR[1] TO WR[0]                         |   |                               |
| U 1393, 17A0,15 | : | 21544 | PORT.WRITE PORT.ADRS[CSR] WITH WR[0]       | : | WRITE THE CSR                 |
|                 | : | 21545 |                                            | : | MAINT=0,CRDY=1,F<2:0>=110     |

U 13

```

21546 -----
21547 -----
21548 *****
21549 * IDC *
21550 *****
21551 -----
21552 018:
21553 =000
21554 DIAG.IDLE:
21555
21556 BEN0/F0,
21557 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
21558 BEN2/F2,
21559 NAD/DIAG.IDLE
21560 -----
21561 01E:
21562 =110
21563
21564 BEN0/CRDY.I,
21565 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
21566 NAD/TEST.FUNC6 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
21567 -----
21568 072:
21569 =0*0
21570 TEST.FUNC6:
21571
21572 UCLR.DSR.0:
21573
21574 UCMD/SET.FIFOA, ;SELECT FIFO A
21575 NAD/UCLR.DSR.1
21576 -----
21577 1E0:
21578 UCLR.DSR.1:
21579
21580 ENB.FIFO/ON, ;ENABLE FIFO A ONTO BUS OUT
21581 LOAD.DSR/ON, ; AND LOAD THE DATA INTO THE DSR
21582 INCR.FIFO.CNTR/ON, ; INCREMENT THE FIFO A CNTR
21583 NAD/UCLR.DSR.2
21584 -----
21585 1E1:
21586 UCLR.DSR.2:
21587
21588 FUNC/CLR.DSR, ;CLEAR THE DSR
21589 NAD/UCLR.DSR.3
21590 -----
21591 1E2:
21592 UCLR.DSR.3:
21593
21594 ENB.DSR/ON, ;LOAD THE DSR ONTO BUS IN
21595 WRT.FIFO/ON, ;WRITE THE DATA INTO FIFO
21596 NAD/XFER.REQ ;SEND XFER REQ TO FLAG THE CPU
21597 -----
21598 1D8:
21599 XFER.REQ:
21600 UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU

```

U 13  
U 13

U 13  
U 13

U 13  
U 13  
U 13  
U 13

U 13  
U 13  
U 13

U 13  
U 13  
U 13

U 13  
U 13  
U 14

U 14  
U 14

U 14  
U 14  
U 14

U 14  
U 14  
U 14  
U 14

U 14

```

:21601 : NAD/DIAG.WAIT
:21602 :-----
:21603 :1D7:
:21604 :DIAG.WAIT:
:21605 :
:21606 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:21607 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:21608 :-----
:21609 :159:
:21610 :=*0*
:21611 :DIAG.WAIT1:
:21612 :
:21613 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:21614 : ; COMMAND FROM THE CPU
:21615 :-----
:21616 :15B:
:21617 :=*1*
:21618 :DIAG.WAIT2:
:21619 :
:21620 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:21621 : ; WAIT SOME MORE
:21622 :-----
:21623 :WAIT.IDC.T24.2:
:21624 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ TO SET
:21625 : JMP [IDC.DONE.T24.2] ;XFER REQ ASSERTED
:21626 : JMP [WAIT.IDC.T24.2] ;WAIT SOME MORE
:21627 :-----
:21628 :IDC.DONE.T24.2:
:21629 : MOV LS[SETCSR.F0] TO WR[0] ;SET UP TO CLR CSR F<2:0>
:21630 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ; WRITE THE CSR
:21631 : MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR XFER REQ
:21632 :-----
:21633 :READ.FIFO.T24.2:
:21634 : MISC [SET.PORT.I.0] ;SELECT THE IDC
:21635 : PORT.READ PORT.ADRS[DATA.BYTE] ;SET UP TO READ THE FIFO
:21636 : MOV PORT TO LS[PORT0] ;READ THE DATA FROM FIFO AND
:21637 : ; SAVE IT IN LS
:21638 : MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
:21639 : CLR WR[1] ;SET UP THE EXPECTED DATA
:21640 :-----
:21641 : JSR [CHECK.RESULT] ;CHECK THE DATA
:21642 : JMP [LOOP.T24.2] ;LOOP HERE IF ERR & LOE IS SET
:21643 :-----
:21644 : JSR [CLEANUP] ;CLEANUP THE CONDITIONS SET
:21645 : ;BY THIS TEST
:21646 : END.T24:
:21647 :
:21648 :

```

U 1394, DB00,1A  
 U 1395, 8939,74  
 U 1396, 8939,44

U 1397, B700,15  
 U 1398, 17A0,15  
 U 1399, 90FA,15

U 139A, 9090,15  
 U 139B, 1788,15  
 U 139C, 3B32,15

U 139D, 3732,15  
 U 139E, 2F82,95

U 139F, 0869,3C  
 U 13A0, 0938,B4

U 13A1, 8875,3C



:21649 PAGE "Test 25 SERIAL SHIFT OUT FUNCTION OF DSR TEST (M8388 IDC MODULE)"  
:21650 ++  
:21651  
:21652 Test Description:  
:21653  
:21654 This test is designed to insure that the Data Shift Register can  
:21655 serially shift out data correctly. The data pattern will be loaded into  
:21656 FIFO A. The DSR will be loaded from FIFO A.  
:21657 The data will be shifted one position to the right and then written  
:21658 into FIFO B. The CPU will then read the data from FIFO B and check.  
:21659 The data patterns that will be used are:  
:21660 Data Pattern 1 = 80  
:21661 Data Pattern 2 = 40  
:21662 Data Pattern 3 = 20  
:21663 Data Pattern 4 = 10  
:21664 Data Pattern 5 = 08  
:21665 Data Pattern 6 = 04  
:21666 Data Pattern 7 = 02  
:21667 Data Pattern 8 = 01  
:21668  
:21669 The expected data will be the data pattern shifted right once.  
:21670  
:21671 Assumptions:  
:21672  
:21673 It is assumed that all previous test have run successfully.  
:21674  
:21675 Test Steps:  
:21676  
:21677 1. Set up error mask, error number and module number in LS  
:21678 (for error printouts) and clear the previous error number in LS.  
:21679 2. Force the IDC microcode to DIAG.IDLE.  
:21680 3. Clear the FIFO B Address Counter.  
:21681 4. Clear the FIFO A Address Counter.  
:21682 5. Write the data pattern into FIFO A.  
:21683 6. Clear the FIFO A Address Counter.  
:21684 7. Set up the CSR to branch to the correct IDC routine.  
:21685  
:21686 \*\*\*\*\*  
:21687 \* IDC \*  
:21688 \*\*\*\*\*  
:21689  
:21690 IDC1. Load the data from FIFO A into the DSR.  
:21691 IDC2. Set UCMD = 6, Change FIFO to select FIFO B and DSR will  
:21692 shift one.  
:21693 IDC3. Enable the data from the DSR through the DSR BUFFER and  
:21694 into FIFO B.  
:21695 IDC4. Issue XFER REQ to flag the CPU.  
:21696  
:21697 8. Wait for a XFER.REQ from IDC.  
:21698 9. Clear CSR F<2:0> and issue XFER GRANT to clear the XFER.REQ.  
:21699 10. Check FIFO B address 0.  
:21700 11. Shift the data pattern and the expected data once to the right and  
:21701 go to Step 3 until all data patterns have been tested.  
:21702 12. Clear IDC.  
:21703 13. End test.

U 14

U 14

U 14

U 14

[illegible]

U 142  
U 142  
U 142  
  
U 143  
U 143  
  
U 143  
  
U 143  
U 143  
U 143  
  
U 143  
U 143  
  
U 143  
  
U 143  
U 143  
U 143  
  
U 143  
U 143  
  
U 143

```

21814 :-----
21815 :1E4:
21816 :SHIFT.DSR.2:
21817 :
21818 : UCMD/CHG.FIFO, ;SELECT FIFO B
21819 : NAD/SHIFT.DSR.3 ;DSR WILL SHIFT ONCE
21820 :-----
21821 :1E5:
21822 :SHIFT.DSR.3:
21823 :
21824 : ENB.DSR/ON, ;LOAD THE DSR ONTO BUS IN
21825 : WRT.FIFO/ON, ;WRITE THE DATA INTO FIFO B
21826 : NAD/XFER.REQ ;SEND XFER REQ TO FLAG THE CPU
21827 :-----
21828 :1D8:
21829 :XFER.REQ:
21830 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
21831 : NAD/DIAG.WAIT
21832 :-----
21833 :1D7:
21834 :DIAG.WAIT:
21835 :
21836 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
21837 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
21838 :-----
21839 :159:
21840 :=*0*
21841 :DIAG.WAIT1:
21842 :
21843 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
21844 : ; COMMAND FROM THE CPU
21845 :-----
21846 :15B:
21847 :=*1*
21848 :DIAG.WAIT2:
21849 :
21850 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
21851 : ; WAIT SOME MORE
21852 :-----
21853 :WAIT.IDC.T25.1:
21854 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ FROM IDC
21855 : JMP [IDC.DONE.T25.1] ;XFER REQ SET
21856 : JMP [WAIT.IDC.T25.1] ;WAIT SOME MORE
21857 :-----
21858 :IDC.DONE.T25.1:
21859 : MOV LS[SETCSR.F0] TO WR[0] ;CLEAR CSR F<2:0>
21860 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
21861 : MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
21862 :-----
21863 :
21864 : PORT.CONTROL [SEL.FIFO.B] ;SELECT FIFO B
21865 : PORT.CONTROL [CLEAR.FIFO.CNTR] ;SET FIFO A ADRS = 0
21866 :-----
21867 :READ.FIFO.T25.1:
21868 : MISC [SET.PORT.I.0] ;SELECT THE IDC

```

U 13BA, DB00,1A  
 U 13BB, 093B,D4  
 U 13BC, 893B,A4

U 13BD, B700,15  
 U 13BE, 17A0,15  
 U 13BF, 90FA,15

U 13C0, 97DC,15  
 U 13C1, 17C0,15

U 13C2, 9090,15

U 14  
U 14  
U 14  
U 14  
U 14  
U 14  
U 14  
U 14

:21887  
:21888  
:21889  
:21890  
:21891  
:21892  
:21893  
:21894  
:21895  
:21896  
:21897  
:21898  
:21899  
:21900  
:21901  
:21902  
:21903  
:21904  
:21905  
:21906  
:21907  
:21908  
:21909  
:21910  
:21911  
:21912  
:21913  
:21914  
:21915  
:21916  
:21917  
:21918  
:21919  
:21920  
:21921  
:21922  
:21923  
:21924  
:21925  
:21926  
:21927  
:21928  
:21929  
:21930  
:21931  
:21932  
:21933  
:21934  
:21935  
:21936  
:21937  
:21938  
:21939  
:21940  
:21941

PAGE "Test 26 RL02 DATA PATH TEST 1 (M8388 IDC MODULE)"  
++  
Test Description:  
  
This test will verify that the IDC produces a disk clock when in the diskless loop mode. It will also verify the RL02 data path on IDC excluding the drivers and receivers to the RL02 Disk Drive using a data pattern of a byte of zeros.  
The data will be loaded into data buffer A. It will then be shifted through the DSR and then through the MFM encoder. The data will then go through the data separator, through the serial input of the DSR and into data buffer A address = 1.  
The CPU will then read and check the data.  
  
Assumptions:  
  
It is assumed that the Data Input, Data Output Buffer, Data Buffer A, Data Buffer B, Bus In, Bus Out, and the parallel inputs and outputs of the DSR are functional.  
  
Test Steps:  
  
1. Set up error mask, error number and module number in LS (for error printouts) and clear the previous error number in LS.  
2. Load the data pattern into Data Buffer A.  
3. Set up the CSR to branch to the correct IDC routine.  
  
\*\*\*\*\*  
\* IDC \*  
\*\*\*\*\*  
  
IDC1. Shift out 32 zeros to sync up data separator.  
IDC2. Load the data pattern byte into the DSR.  
IDC3. Shift the byte thru the MFM encoder and back thru the data separator into the DSR.  
IDC4. Write the byte into Data Buffer A.  
IDC5. Issue a XFER.REQ to the CPU to indicate IDC has completed the operation.  
  
4. Wait for XFER REQ from the IDC.  
5. If XFER REQ does not assert then issue error to indicate no response from the IDC.  
6. Increment the error number.  
7. Read Data Buffer A address = 1 and check the data.  
8. End test.  
  
Errors:  
  
ERROR 1 - One or more of the clocks stopped when the Disk Clock was selected in conjunction with the RL02 diskless loop.  
  
Printout:  
  

EXPECTED

RECEIVED

N/A
N/A

U 14  
U 14  
U 14

U 14  
U 14  
U 14

U 14  
U 14  
U 14  
U 14

U 14  
U 14

U 14  
U 14  
U 14  
U 14

U 14  
U 14

|              |   |           |         |           |             |                    |                                       |              |          |
|--------------|---|-----------|---------|-----------|-------------|--------------------|---------------------------------------|--------------|----------|
| ZZ-ENKCF-1.4 | : | ENKCF.MIC | Test 26 | RL02 DATA | 15-MAR-83   | Fiche 3            | Frame E4                              | Sequence 455 |          |
| :            | : | ENKCF.MCR | MICRO2  | 1M(01)    | 15-MAR-83   | 11:46:22           | ENKCF Micro-diagnostic for IDC module | Rev 01.40    | Page 449 |
| :            | : | ENKCF.MIC | Test 26 | RL02 DATA | PATH TEST 1 | (M8388 IDC MODULE) |                                       |              |          |

  

|                 |   |       |   |                                                                   |                                |
|-----------------|---|-------|---|-------------------------------------------------------------------|--------------------------------|
|                 | : | 21997 | : | The data will then be loaded into Data Buffer A address = 1 to be |                                |
|                 | : | 21998 | : | read and checked by the CPU.                                      |                                |
|                 | : | 21999 | : | --                                                                |                                |
|                 | : | 22000 | : | *****                                                             |                                |
|                 | : | 22001 | : | T.26:                                                             |                                |
| U 13CD, B65E,15 | : | 22002 | : | MOV LS[BEGIN.TEST] TO WR[0]                                       | ;SET BIT 15 IN WR[0] FOR       |
|                 | : | 22003 | : |                                                                   | ; CONTROL AND STATUS WORD      |
| U 13CE, 3E80,15 | : | 22004 | : | MOV WR[0] TO LS[CONTROL.STATUS]                                   | ;SET BIT 15 IN CONTROL AND     |
|                 | : | 22005 | : |                                                                   | ; STATUS WORD - BIT 15         |
|                 | : | 22006 | : |                                                                   | ; INDICATES START OF TEST TO   |
|                 | : | 22007 | : |                                                                   | ; CONSOLE                      |
| U 13CF, 10E0,15 | : | 22008 | : | MISC [SET.CP.ATTN]                                                | ;ISSUE CPU ATTN TO CONSOLE     |
|                 | : | 22009 | : | WAIT.T26.0:                                                       |                                |
| U 13D0, 893D,04 | : | 22010 | : | JMP [WAIT.T26.0]                                                  | ;LOOP HERE WAITING FOR         |
|                 | : | 22011 | : |                                                                   | ; CONSOLE TO RESPOND           |
| U 13D1, 65A0,15 | : | 22012 | : | CLR LS[PREVIOUS.ERROR]                                            | ;CLEAR PREVIOUS ERROR NUMBER   |
|                 | : | 22013 | : |                                                                   | ; IN LS                        |
|                 | : | 22014 | : |                                                                   |                                |
|                 | : | 22015 | : |                                                                   |                                |
| U 13D2, E58A,15 | : | 22016 | : | CLR LS[ERROR.MASK]                                                | ;CHECK ALL BITS                |
|                 | : | 22017 | : |                                                                   |                                |
| U 13D3, B66E,15 | : | 22018 | : | MOV LS[NA.EXP.REC] TO WR[0]                                       | ;SET UP TO PRINT N/A FOR       |
| U 13D4, 3E80,15 | : | 22019 | : | MOV WR[0] TO LS[CONTROL.STATUS]                                   | ; EXPECTED & RECEIVED DATA     |
|                 | : | 22020 | : |                                                                   |                                |
| U 13D5, B64A,15 | : | 22021 | : | MOV LS[IDC] TO WR[0]                                              | ;STORE MODULE CODE IN LS TO    |
| U 13D6, 3E8C,15 | : | 22022 | : | MOV WR[0] TO LS[MODULE.NUM]                                       | ; INDICATE IDC MODULE          |
|                 | : | 22023 | : |                                                                   |                                |
| U 13D7, B640,15 | : | 22024 | : | MOV LS[#1] TO WR[0]                                               |                                |
| U 13D8, BE82,15 | : | 22025 | : | MOV WR[0] TO LS[ERROR.NUMBER]                                     | ;SET UP ERROR NUMBER = 1 IN LS |
|                 | : | 22026 | : |                                                                   |                                |
|                 | : | 22027 | : |                                                                   |                                |
| U 13D9, B78E,15 | : | 22028 | : | MOV LS[DRIVE.STATUS] TO WR[0]                                     | ;HAS THE DRIVE SIZING ROUTINE  |
|                 | : | 22029 | : | BIT LS[BIT7] WITH WR[0],                                          | ; BEEN RUN?                    |
| U 13DA, D94E,35 | : | 22030 | : | DT(LONG)&SET.ALU.CC                                               |                                |
| U 13DB, 093D,D1 | : | 22031 | : | JMP.IF[BIT.SET] TO [SIZE.OK.T26.1]                                | ;IF YES THEN GO RUN TEST       |
|                 | : | 22032 | : |                                                                   |                                |
| U 13DC, 8870,0C | : | 22033 | : | JSR [DRIVE.SIZE]                                                  | ;GO RUN THE SIZING ROUTINE     |
|                 | : | 22034 | : |                                                                   |                                |
|                 | : | 22035 | : | SIZE.OK.T26.1:                                                    |                                |
| U 13DD, 8872,EC | : | 22036 | : | JSR [DIAG.CODE]                                                   | ;FORCE THE IDC TO DIAG.IDLE    |
|                 | : | 22037 | : |                                                                   |                                |
|                 | : | 22038 | : | ;SELECT AN RL02 DRIVE                                             |                                |
|                 | : | 22039 | : |                                                                   |                                |
| U 13DE, B78E,15 | : | 22040 | : | MOV LS[DRIVE.STATUS] TO WR[0]                                     |                                |
|                 | : | 22041 | : | BIT LS[BIT0] WITH WR[0],                                          | ;IS DRIVE 0 AN R80?            |
| U 13DF, 5940,35 | : | 22042 | : | DT(LONG)&SET.ALU.CC                                               |                                |
| U 13E0, 093E,39 | : | 22043 | : | JMP.IF[BIT.CLR] TO [SET.DS0.T26.1]                                | ;JMP IF DRV 0 IS NOT AN R80    |
| U 13E1, B717,15 | : | 22044 | : | MOV LS[SETCSR.DS1] TO WR[2]                                       | ;SAVE THE DATA TO SEL DRIVE 1  |
| U 13E2, 093E,44 | : | 22045 | : | JMP [TEST.T26.1]                                                  |                                |
|                 | : | 22046 | : | SET.DS0.T26.1:                                                    |                                |
| U 13E3, 3715,15 | : | 22047 | : | MOV LS[SETCSR.DS0] TO WR[2]                                       | ;SAVE THE DATA TO SEL DRIVE 0  |
|                 | : | 22048 | : |                                                                   |                                |
|                 | : | 22049 | : | TEST.T26.1:                                                       |                                |
|                 | : | 22050 | : | LOOP.T26.1:                                                       |                                |
|                 | : | 22051 | : | LOOP.T26.2:                                                       |                                |



```

U 13E4, 2F80,15 :22052 CLR WR[0] ;SET DATA = 0
U 13E5, 17D8,15 :22053 PORT.CONTROL [SEL.FIFO.A] ;SELECT FIFO A FOR WRITE
U 13E6, 17C0,15 :22054 PORT.CONTROL [CLEAR.FIFO.CNTR] ;SET FIFO TO ADDRESS 0
U 13E7, 97A8,15 :22055 PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;WRITE THE DATA TO FIFO A
 :22056
U 13E8, B626,15 :22057 MOV LS[0FF] TO WR[0] ;WRITE FIFO A ADDRESS 1 = FF
U 13E9, 97A8,15 :22058 PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;WRITE THE DATA TO FIFO A
 :22059
 :22060
U 13EA, 17C0,15 :22061 PORT.CONTROL [CLEAR.FIFO.CNTR] ;SET FIFO TO ADDRESS 0
U 13EB, 3708,15 :22062 MOV LS[SETCSR.F4] TO WR[0] ;SET CSR F<2:0>=100 TO BRANCH
 :22063
 :22064
U 13EC, AEC4,15 :22064 BIS WR[2] TO WR[0] ;INCLUDE THE RLO2 DRV SELECT
U 13ED, 17A0,15 :22065 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE DATA TO IDC CSR
 :22066
 :22067
 :22068
 :22069 *****
 :22070 * IDC *
 :22071 *****
 :22072
 :22073 018:
 :22074 =000
 :22075 DIAG.IDLE:
 :22076
 :22077 BEN0/F0,
 :22078 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
 :22079 BEN2/F2,
 :22080 NAD/DIAG.IDLE
 :22081
 :22082 01C:
 :22083 =100
 :22084
 :22085 BEN0/CRDY.L,
 :22086 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
 :22087 NAD/TEST.FUNC4 ;DEPENDENT ON CSR <CRDY> AND <MAINT>
 :22088
 :22089 06B:
 :22090 =0+1
 :22091 DISKLESS.DIAG.0:
 :22092
 :22093 UMAINT/ON,
 :22094 LOOP.LOCK/ON, ;MAINT PREVENTS LOOP LOCK FROM SELECTING
 :22095 DISK.WRITE/ON, ;READ DATA...DISK WRITE STILL SELECTS DSRO
 :22096 LOAD.LOOP.CNTR.[0E0],
 :22097 NAD/DISKLESS.DIAG.1
 :22098
 :22099 184:
 :22100 =0
 :22101 DISKLESS.DIAG.1:
 :22102
 :22103 UMAINT/ON,
 :22104 LOOP.LOCK/ON, ;SHIFT OUT 32 ZEROS TO SYNC UP
 :22105 DISK.WRITE/ON, ;DATA SEPARATOR
 :22106 UCMD/SET.FIFOA,

```

```

:22107 : INCR.LPCNTR/ON,
:22108 : BENO/CNTR.OVFLW,
:22109 : NAD/DISKLESS.DIAG.1
:22110 : -----
:22111 : 185:
:22112 : =1
:22113 : DISKLESS.DIAG.2:
:22114 :
:22115 : UMAINT/ON,
:22116 : LOOP.LOCK/ON,
:22117 : DISK.WRITE/ON, ;MUST WAIT 3 CYCLES FOR SYNC SEEN
:22118 : ENB.DSK.CLK,
:22119 : NAD/DISKLESS.DIAG.3A
:22120 : -----
:22121 : 1D9:
:22122 : DISKLESS.DIAG.3A:
:22123 :
:22124 : UMAINT/ON,
:22125 : LOOP.LOCK/ON,
:22126 : DISK.WRITE/ON,
:22127 : NAD/DISKLESS.DIAG.3B
:22128 : -----
:22129 : 1DA:
:22130 : DISKLESS.DIAG.3B:
:22131 :
:22132 : UMAINT/ON,
:22133 : LOOP.LOCK/ON,
:22134 : DISK.WRITE/ON,
:22135 : NAD/DISKLESS.DIAG.3
:22136 : -----
:22137 : 1DB:
:22138 : DISKLESS.DIAG.3:
:22139 :
:22140 : UMAINT/ON,
:22141 : LOOP.LOCK/ON,
:22142 : DISK.WRITE/ON, ;LOAD THE DATA FROM FIFO A INTO THE DSR
:22143 : ENB.FIFO/ON,
:22144 : LOAD.DSR/ON,
:22145 : NAD/DISKLESS.DIAG.4A
:22146 : -----
:22147 : 1DC:
:22148 : DISKLESS.DIAG.4A:
:22149 :
:22150 : UMAINT/ON,
:22151 : LOOP.LOCK/ON, ;INCREMENT THE FIFO ADDRESS COUNTER
:22152 : DISK.WRITE/ON,
:22153 : INCR.FIFO.CNTR/ON,
:22154 : LOAD.LOOP.CNTR.[OF5],
:22155 : NAD/DISKLESS.DIAG.4
:22156 : -----
:22157 : 188:
:22158 : =0
:22159 : DISKLESS.DIAG.4:
:22160 :
:22161 : UMAINT/ON,

```

U 14  
 U 14  
 U 14

U 14  
 U 14  
 U 14

U 14  
 U 14  
 U 14

U 14  
 U 14  
 U 14  
 U 14

U 14

```

:22162 : LOOP.LOCK/ON,
:22163 : DISK.WRITE/ON, ;WAIT FOR DATA TO SHIFT THRU DISKLESS
:22164 : INCR.LPCNTR/ON, ; LOOP
:22165 : BEN0/CNTR.OVFLW,
:22166 : NAD/DISKLESS.DIAG.4
:-----
:22167 :
:22168 :189:
:22169 : =1
:22170 : DISKLESS.DIAG.5:
:22171 :
:22172 : UMAINT/ON,
:22173 : LOOP.LOCK/ON,
:22174 : ENB.DSR/ON, ;LOAD THE DATA INTO FIFO A
:22175 : WRT.FIFO/ON,
:22176 : DISK.WRITE/ON,
:22177 : NAD/DISKLESS.DIAG.6
:-----
:22178 :
:22179 :1DD:
:22180 : DISKLESS.DIAG.6:
:22181 :
:22182 : UMAINT/ON,
:22183 : LOOP.LOCK/ON,
:22184 : DISK.WRITE/ON,
:22185 : ENB.CPU.CLK, ;SELECT THE CPU CLK
:22186 : NAD/XFER.REQ ;SEND XFER REQ TO FLAG THE CPU
:-----
:22187 :
:22188 :1D8:
:22189 : XFER.REQ:
:22190 :
:22191 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:22192 : NAD/DIAG.WAIT
:-----
:22193 :
:22194 :1D7:
:22195 : DIAG.WAIT:
:22196 :
:22197 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:22198 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:-----
:22199 :
:22200 :159:
:22201 : =*0*
:22202 : DIAG.WAIT1:
:22203 :
:22204 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:22205 : ; COMMAND FROM THE CPU
:-----
:22206 :
:22207 :15B:
:22208 : =*1*
:22209 : DIAG.WAIT2:
:22210 :
:22211 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:22212 : ; WAIT SOME MORE
:-----
:22213 :
:22214 :
:22215 : MOV LSE[40] TO WR[0] ;SET THE LOOP COUNT
:22216 : WAIT.IDC.T26:

```

```

U 13EF, DB00,1A :22217 SKIP.IF[NO.FAST.INTERRUPT] ;SKIP UNTIL IDC DONE
U 13F0, 893F,74 :22218 JMP [IDC.DONE.T26] ;EXIT LOOP WHEN IDC IS DONE
:22219 DEC WR[0], ;SET UP A LOOP TO CATCH
U 13F1, A100,35 :22220 DT(LONG)&SET.ALU.CC ; IF THE SEQ STOPS WITH THE
U 13F2, 893E,F1 :22221 JMP.IF[NEQ] TO [WAIT.IDC.T26] ; DISK CLOCK SELECTED
:22222
U 13F3, 2F80,15 :22223 CLR WR[0] ;FORCE AN ERROR TO REPORT THAT
U 13F4, 3640,95 :22224 MOV LS[#1] TO WR[1] ; THE IDC IS NOT RESPONDING IN
U 13F5, 0869,3C :22225 JSR [CHECK.RESULT] ; THE DISKLESS LOOP
U 13F6, 093E,44 :22226 JMP [LOOP.T26.1] ;LOOP HERE IF ERR & LOE IS SET
:22227
:22228 IDC.DONE.T26:
:22229
U 13F7, 2F80,15 :22230 CLR WR[0]
U 13F8, 2F82,95 :22231 CLR WR[1]
U 13F9, 0869,3C :22232 JSR [CHECK.RESULT] ;CALL CHECK.RESULT TO LOCK
:22233 ; IN INTERMITTANT ERRORS
U 13FA, 093E,44 :22234 JMP [LOOP.T26.1]
:22235
U 13FB, B700,15 :22236 MOV LS[SETCSR.F0] TO WR[0] ;CLEAR THE FUNC BITS
U 13FC, 17A0,15 :22237 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ; WRITE THE DATA TO IDC CSR
U 13FD, 90FA,15 :22238 MISC2 [TRANSFER.GRANT] ;SEND TRANSFER GRANT
:22239
U 13FE, 3642,15 :22240 MOV LS[#2] TO WR[0]
U 13FF, BE82,15 :22241 MOV WR[0] TO LS[ERROR.NUMBER] ;ERROR 2
U 1400, E580,15 :22242 CLR LS[CONTROL.STATUS] ;ALLOW EXP AND REC DATA TO
:22243 ; BE PRINTED ON ERROR
U 1401, B62C,15 :22244 MOV LS[FFFFFFF0] TO WR[0]
U 1402, 3E8A,15 :22245 MOV WR[0] TO LS[ERROR.MASK] ;SET MASK TO CHECK LO BYTE
:22246
:22247 ;READ THE BYTE OF DATA
U 1403, 9090,15 :22248 MISC [SET.PORT.1.0] ;SELECT THE IDC
U 1404, 1788,15 :22249 PORT.READ PORT.ADRS[DATA.BYTE] ;SET UP TO READ BYTE OF DATA
U 1405, 3B32,15 :22250 MOV PORT TO LS[PORT0] ;READ THE DATA FROM FIFO AND
:22251 ; SAVE IT IN LS
U 1406, 3732,15 :22252 MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
U 1407, 2F82,95 :22253 CLR WR[1] ;SET UP THE EXPECTED DATA
U 1408, 0869,3C :22254 JSR [CHECK.RESULT] ;CHECK THE DATA
U 1409, 093E,44 :22255 JMP [LOOP.T26.2] ;LOOP HERE IF ERR & LOE IS SET
:22256
:22257
U 140A, 8875,3C :22258 JSR [CLEANUP] ;CLEANUP CONDITIONS SET BY
:22259 ; THIS TEST
:22260
:22261 END.T26:
:22262
:22263
:22264

```

:22265 :PAGE "Test 27 RL02 DATA PATH TEST 2 (M8388 IDC MODULE)"

:22266 :++

:22267 :

:22268 :Test Description:

:22269 :

:22270 :This test will verify the RL02 data path on IDC excluding the

:22271 :the drivers and receivers to the RL02 Disk Drive. The data will be

:22272 :loaded into data buffer A. It will then be shifted through the DSR and

:22273 :then through the MFM encoder. The data will then go through the data

:22274 :separator, through the serial input of the DSR and into data buffer A

:22275 :ADDRESS =1.

:22276 :The CPU will then read data buffer A and check the data.

:22277 :The data patterns that will be used are:

:22278 :1. AA

:22279 :2. 55

:22280 :3. 33

:22281 :4. 0F

:22282 :5. FF

:22283 :

:22284 :Assumptions:

:22285 :

:22286 :It is assumed that the Data Input, Data Output Buffer, Data

:22287 :Buffer A, Bus In, Bus Out, and the parallel inputs and outputs

:22288 :of the DSR are functional.

:22289 :

:22290 :Test Steps:

:22291 :

:22292 :1. Set up error mask, error number and module number in LS

:22293 : (for error printouts) and clear the previous error number in LS.

:22294 :2. Load the data pattern into Data Buffer A from WR[0].

:22295 :3. Set up the CSR to branch to the correct IDC routine.

:22296 :

:22297 :

:22298 :\*\*\*\*\*

:22299 :\* IDC \*

:22300 :\*\*\*\*\*

:22301 :

:22302 :IDC1. Shift out 32 zeros to sync up data separator.

:22303 :IDC2. Load the data pattern byte into the DSR.

:22304 :IDC3. Shift the byte thru the MFM encoder and back thru the

:22305 :data separator into the DSR.

:22306 :IDC4. Write the byte into Data Buffer A address = 1.

:22307 :IDC5. Issue a XFER.REQ to the CPU to indicate IDC has completed

:22308 :the operation.

:22309 :

:22310 :4. Check the contents of FIFO A, ADDRESS 1. If incorrect,

:22311 :issue ERROR.

:22312 :Else load next data pattern into WR[0] and GO TO Step 2 until all

:22313 :data patterns have been tested.

:22314 :

:22315 :Errors:

:22316 :

:22317 :

:22318 :

:22319 :

Printout:

EXPECTED  
FIFO A

RECEIVED  
FIFO A

U 14

U 14

U 14

U 14

U 14

U 14

U 14

U 14

U 14

U 14

U 14

U 14

U 14

U 14

U 14

U 14

U 14

U 14

:22320 : ADDR=1 ADDR=1  
:22321 :  
:22322 : ERROR 1 - RL02 Data Path failed using data pattern of AA  
:22323 : ERROR 2 - RL02 Data Path failed using data pattern of 55  
:22324 : ERROR 3 - RL02 Data Path failed using data pattern of 33  
:22325 : ERROR 4 - RL02 Data Path failed using data pattern of 0F  
:22326 : ERROR 5 - RL02 Data Path failed using data pattern of FF  
:22327 :  
:22328 : Debug:  
:22329 :  
:22330 : The RL02 Data Path up to the parallel load of the DSR has been proven  
:22331 : fully operational in previous tests. After the data is parallel loaded  
:22332 : into the DSR, the control line LOAD DSR H should deassert and the data  
:22333 : should then be serially shifted out through DSR 0. If this is not  
:22334 : happening correctly, then either the control line LOAD DSR H did not  
:22335 : deassert, or DSR 0 is shorted or the DSR itself is bad.  
:22336 :  
:22337 : The data is then shifted into the MFM ENCODER PAL to become  
:22338 : WRT DATA <3:0>.  
:22339 : Input UMAINT H should be H and 4.1 MHZ CLK L should be clocking  
:22340 : DSR 0 through to WRT DATA <3:0>. The 4.1 MHZ CLK is generated by a  
:22341 : 8.2 MHZ crystal oscillator located on the IDC module. The 8.2 MHZ  
:22342 : is then divided through the PEAK SHIFT COMP PAL to become 4.1 MHZ.  
:22343 :  
:22344 : WRT DATA <3:0> then becomes the inputs to the PEAK SHIFT COMP PAL.  
:22345 : This PAL modifies the frequency of the write data which would in  
:22346 : normal operation be the data written on the disk. However in this  
:22347 : diskless loop, the data is output as WRITE DATA L, and then  
:22348 : immediately looped back through the data separator.  
:22349 :  
:22350 : The data is output from the Data Separator as DS DATA H. DS DATA H is  
:22351 : then ANDed with SEL STATUS CLK L which should be deasserted, and the  
:22352 : data becomes read data RL DATA L.  
:22353 :  
:22354 : RL DATA L is then selected through a MIX by RL02 H = H, and becomes  
:22355 : READ DATA H. READ DATA H is then latched up through a flip flop and  
:22356 : becomes SER DATA IN H which is the serial input to the DSR. After the  
:22357 : byte of data has been serially shifted into the DSR, the byte of data  
:22358 : will be loaded into a quad buffer which is enabled by UENB DSR L = L.  
:22359 :  
:22360 : The data will then be loaded into Data Buffer A address = 1, to  
:22361 : be read and checked by the CPU.  
:22362 : --  
:22363 : \*\*\*\*\*  
:22364 : T.27:

U 140B, B65E,15 :22365 : MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT 15 IN WR[0] FOR  
:22366 : ; CONTROL AND STATUS WORD  
U 140C, 3E80,15 :22367 : MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND  
:22368 : ; STATUS WORD - BIT 15  
:22369 : ; INDICATES START OF TEST TO  
:22370 : ; CONSOLE  
U 140D, 10E0,15 :22371 : MISC [SET.CP.ATTN] ;ISSUE CPU ATTN TO CONSOLE  
:22372 : WAIT.T27.0:  
U 140E, 0940,E4 :22373 : JMP [WAIT.T27.0] ;LOOP HERE WAITING FOR  
:22374 : ; CONSOLE TO RESPOND

U 14  
U 14  
U 14  
  
U 14  
U 14

U 149

11 149

U 149

U 149

U 149

U 149

U 149

U 149

6.14.

11 145

U 149

11 149

|    |     |
|----|-----|
| 0  | 147 |
| 11 | 149 |

11 164

11 164

0 147

1

11 161

U 141

U 141

U 141

U 141

U 141

```

:22430 LOOP.T27.3:
U 142D, B6C4,15 :22431 MOV LS[#33333333] TO WR[0] ;GET THE NEXT DATA PATTERN
U 142E, 3F24,15 :22432 MOV WR[0] TO LS[SAV.DATA.PAT] ;SAVE IT
U 142F, 0873,DC :22433 JSR [WRITE.READ.BYTE] ;CYCLE BYTE OF [33] THRU RL02
:22434 ; DATA PATH
U 1430, 0869,3C :22435 JSR [CHECK.RESULT] ;GO CHECK RESULT
U 1431, 8942,D4 :22436 JMP [LOOP.T27.3] ;LOOP HERE IF ERR & LOE IS SET
:22437
:22438
U 1432, FF82,15 :22439 INC LS[ERROR.NUMBER] ;ERROR 4
:22440
:22441 LOOP.T27.4:
U 1433, 36C6,15 :22442 MOV LS[#0F0F0F0F] TO WR[0] ;GET THE NEXT DATA PATTERN
U 1434, 3F24,15 :22443 MOV WR[0] TO LS[SAV.DATA.PAT] ;SAVE IT
U 1435, 0873,DC :22444 JSR [WRITE.READ.BYTE] ;CYCLE BYTE OF [0F] THRU RL02
:22445 ; DATA PATH
U 1436, 0869,3C :22446 JSR [CHECK.RESULT] ;GO CHECK RESULT
U 1437, 8943,34 :22447 JMP [LOOP.T27.4] ;LOOP HERE IF ERR & LOE IS SET
:22448
U 1438, FF82,15 :22449 INC LS[ERROR.NUMBER] ;ERROR 5
:22450
:22451 LOOP.T27.5:
U 1439, B69E,15 :22452 MOV LS[#FFFFFFF] TO WR[0] ;GET THE NEXT DATA PATTERN
U 143A, 3F24,15 :22453 MOV WR[0] TO LS[SAV.DATA.PAT] ;SAVE IT
U 143B, 0873,DC :22454 JSR [WRITE.READ.BYTE] ;CYCLE BYTE OF [FF] THRU RL02
:22455 ; DATA PATH
U 143C, 0869,3C :22456 JSR [CHECK.RESULT] ;GO CHECK RESULT
U 143D, 8943,94 :22457 JMP [LOOP.T27.5] ;LOOP HERE IF ERR & LOE IS SET
:22458
:22459
U 143E, 8875,3C :22460 JSR [CLEANUP] ;CLEANUP THE CONDITIONS SET BY
:22461 ; THIS TEST
:22462
:22463 END.T27: ;END TEST
:22464
:22465
:22466

```





```

:22522 :
:22523 : EXPECTED RECEIVED
:22524 : ECC STAT<1:0> ECC STAT<1:0>
:22525 : CSR <21:20> CSR <21:20>
:22526 :
:22527 : ERROR 1 - The ECC STAT <1:0> bits did not both initialize to 00.
:22528 :
:22529 : ERROR 2 - DC007 ERR did not set or the ECC STAT<1:0> bits failed.
:22530 :
:22531 : Debug:
:22532 :
:22533 : ERROR 1 -
:22534 : The IDC microcode will execute an instruction with FUNC <3:0>=0001.
:22535 : These bits will be decoded and will assert UCLR ECC L. This signal
:22536 : is the Clear line to the CRC/ECC chip and will deassert ECC STAT 1 H
:22537 : and ECC STAT 0 H. These signals are driven onto IO BUS <21:20> when the
:22538 : CSR is read.
:22539 :
:22540 : ERROR 2 -
:22541 : A one is loaded into FIFO A address 0. This data is then loaded into
:22542 : the DSR. The IDC microword will then assert UMAINT H, UCHECK FIELD H
:22543 : and UENB CRC H. UMAINT H allows DSR 0 H to become NRZ WRT DATA H at the
:22544 : MFM ENCODER PAL. UCHECK FIELD H = H and UENB CRC H = H allow NRZ WRT
:22545 : DATA H to be shifted in to the internal Pattern register of the CRC/ECC
:22546 : chip. In the next IDC micro-instruction, UCHECK FIELD H will be
:22547 : deasserted to make the CRC/ECC chip believe that it has read the check
:22548 : field. UDATA FIELD H will be asserted to keep the internal clock
:22549 : running and ENB CRC H will be asserted to prevent certain CRC/ECC chip
:22550 : internal signals. UMAINT H is still asserted to allow DSR 0 H as
:22551 : NRZ WRT DATA H. When UCHECK FIELD H was deasserted, the internal
:22552 : CRC/ECC chip Pattern register was non-zero so an internal error
:22553 : condition will result, forcing ECC STAT 1 H = L and ECC STAT 0 H = H.
:22554 : These signals are driven onto IO BUS <21:20> when the CSR is read.
:22555 : --
:22556 : *****
:22557 : T.28:
:22558 : MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT 15 IN WR[0] FOR
:22559 : MOV WR[0] TO LS[CONTROL.STATUS] ;CONTROL AND STATUS WORD
:22560 : ;SET BIT 15 IN CONTROL AND
:22561 : ;STATUS WORD - BIT 15
:22562 : ;INDICATES START OF TEST TO
:22563 : ;CONSOLE
:22564 : MISC [SET.CP.ATTN] ;ISSUE CPU ATTN TO CONSOLE
:22565 : WAIT.T28.0:
:22566 : JMP [WAIT.T28.0] ;LOOP HERE WAITING FOR
:22567 : ;CONSOLE TO RESPOND
:22568 :
:22569 : CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
:22570 : ;IN LS
:22571 :
:22572 : MOV LS[IDC] TO WR[0] ;STORE MODULE CODE IN LS TO
:22573 : MOV WR[0] TO LS[MODULE.NUM] ;INDICATE IDC MODULE
:22574 :
:22575 :
:22576 : MOV LS[#FFCFFFFFF] TO WR[0] ;MASK TO CHECK BITS <21:20>

```

```

U 1447, 3E8A,15 :22577 MOV WR[0] TO LS[ERROR.MASK]
:22578
U 1448, 8872,EC :22579 JSR [DIAG.CODE] ;FORCE THE IDC TO DIAG.IDLE
:22580
:22581 LOOP.T28.1:
:22582 LOOP.T28.2:
U 1449, B640,15 :22583 MOV LS[#1] TO WR[0] ;SET WRO = 1
U 144A, BE82,15 :22584 MOV WR[0] TO LS[ERROR.NUMBER] ;SET UP ERROR NUMBER = 1 IN LS
:22585
U 144B, B712,15 :22586 MOV LS[SETCSR.CRDY] TO WR[0] ;SET UP THE CSR TO BRANCH TO
U 144C, B720,95 :22587 MOV LS[SETCSR.MAINT] TO WR[1] ; THE CORRECT IDC ROUTINE
U 144D, AEC2,15 :22588 BIS WR[1] TO WR[0]
U 144E, 370C,95 :22589 MOV LS[SETCSR.F6] TO WR[1]
U 144F, AEC2,15 :22590 BIS WR[1] TO WR[0]
U 1450, 17A0,15 :22591 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:22592 ;MAINT=1,CRDY=1,F<2:0>=110
:22593 -----
:22594 -----
:22595 *****
:22596 * IDC *
:22597 *****
:22598 -----
:22599 :018:
:22600 :=000
:22601 :DIAG.IDLE:
:22602
:22603 BEN0/F0,
:22604 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:22605 BEN2/F2,
:22606 NAD/DIAG.IDLE
:22607 -----
:22608 :01E:
:22609 :=110
:22610
:22611 BEN0/CRDY.L,
:22612 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:22613 NAD/TEST.FUNC6 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:22614 -----
:22615 :076:
:22616 :=1*0
:22617 :TEST.ECC.1:
:22618
:22619 FUNC/CLR.ECC, ;ISSUE CLR ECC TO INIT THE DC007
:22620 NAD/XFER.REQ ;FLAG THE CPU
:22621 -----
:22622 :1D8:
:22623 XFER.REQ:
:22624 UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:22625 NAD/DIAG.WAIT
:22626 -----
:22627 :1D7:
:22628 :DIAG.WAIT:
:22629
:22630 BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:22631 NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE

```

ZZ-1  
 :  
 :  
 U 1  
 U 1

```

:22632 :-----
:22633 :159:
:22634 :=*0*
:22635 :DIAG.WAIT1:
:22636 :
:22637 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:22638 : ; COMMAND FROM THE CPU
:22639 :-----
:22640 :15B:
:22641 :=*1*
:22642 :DIAG.WAIT2:
:22643 :
:22644 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:22645 : ; WAIT SOME MORE
:22646 :-----
:22647 :WAIT.IDC.T28.1:
U 1451, DB00,1A :22648 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ FROM IDC
U 1452, 0945,44 :22649 : JMP [IDC.DONE.T28.1] ;XFER REQ SET
U 1453, 0945,14 :22650 : JMP [WAIT.IDC.T28.1] ;WAIT SOME MORE
:22651 :
:22652 :IDC.DONE.T28.1:
U 1454, B700,15 :22653 : MOV LS[SETCSR.F0] TO WR[0] ;CLEAR CSR F<2:0>
U 1455, 17A0,15 :22654 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
U 1456, 90FA,15 :22655 : MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
:22656 :
:22657 :
:22658 :READ.CSR.T28.1:
U 1457, 9090,15 :22659 : MISC [SET.PORT.I.0] ;SELECT THE IDC
U 1458, 9780,15 :22660 : PORT.READ PORT.ADRS[CSR] ;SET UP TO READ THE CSR
U 1459, 3B32,15 :22661 : MOV PORT TO LS[PORT0] ;READ THE CSR AND
:22662 : ; SAVE IT IN LS
U 145A, 3732,15 :22663 : MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
U 145B, 2F82,95 :22664 : CLR WR[1] ;EXPECTED DATA
U 145C, 0869,3C :22665 : JSR [CHECK.RESULT] ;CHECK THE DATA
U 145D, 0944,94 :22666 : JMP [LOOP.T28.1] ;LOOP HERE IF ERR & LOE IS SET
:22667 :
:22668 :;FORCE AN ERROR ON THE DC007 CHIP AND CHECK ECC STAT<1:0>=01
:22669 :
:22670 :;WRITE A ONE TO THE FIFO TO LATER BE LOADED INTO THE DC007 PATTERN REG TO
:22671 :; FORCE ERROR
:22672 :
U 145E, 3642,15 :22673 : MOV LS[#2] TO WR[0]
U 145F, BE82,15 :22674 : MOV WR[0] TO LS[ERROR.NUMBER] ;ERROR 2
:22675 :
U 1460, 17D8,15 :22676 : PORT.CONTROL [SEL.FIFO.A] ;SELECT FIFO A
U 1461, 17C0,15 :22677 : PORT.CONTROL [CLEAR.FIFO.CNTR] ;SET ADRS = 0
U 1462, B640,15 :22678 : MOV LS[#1] TO WR[0] ;GET THE DATA TO WRITE
U 1463, 97A8,15 :22679 : PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;WRITE A ONE TO FIFO A
U 1464, 17C0,15 :22680 : PORT.CONTROL [CLEAR.FIFO.CNTR] ;SET ADRS = 0
:22681 :
U 1465, 3712,95 :22682 : MOV LS[SETCSR.CRDY] TO WR[1] ;SET UP THE CSR TO BRANCH TO
U 1466, 370E,15 :22683 : MOV LS[SETCSR.F7] TO WR[0] ; THE CORRECT IDC ROUTINE
U 1467, AEC2,15 :22684 : BIS WR[1] TO WR[0]
U 1468, 17A0,15 :22685 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:22686 : ;MAINT=0,CRDY=1,F<2:0>=111

```

```

:22687
:22688 :CONTINUE BRANCHING
:22689
U 1469, B79F,15 :22690 MOV LS[SETCSR.R80] TO WR[2] ;CONTINUE TO BRANCH TO THE
U 146A, 3702,15 :22691 MOV LS[SETCSR.F1] TO WR[0] ; IDC ROUTINE
U 146B, 3712,95 :22692 MOV LS[SETCSR.CRDY] TO WR[1]
U 146C, AEC2,15 :22693 BIS WR[1] TO WR[0] ;SET CRDY TO HOLD SDIO LO
U 146D, AEC4,15 :22694 BIS WR[2] TO WR[0]
U 146E, 17A0,15 :22695 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:22696 ;R80=1,F<2:0>=001
:22697 -----
:22698 -----
:22699 *****
:22700 * IDC *
:22701 *****
:22702 -----
:22703 018:
:22704 =000
:22705 DIAG.IDLE:
:22706
:22707 BEN0/F0,
:22708 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:22709 BEN2/F2,
:22710 NAD/DIAG.IDLE
:22711 -----
:22712 01F:
:22713 =111
:22714
:22715 BEN0/CRDY.L,
:22716 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:22717 NAD/TEST.FUNC7 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:22718 -----
:22719 078:
:22720 =0*0
:22721 TEST.FUNC7:
:22722
:22723 NAD/GROUP0
:22724 -----
:22725 163:
:22726 =0**
:22727 GROUP0:
:22728
:22729 BEN2/R80.FORMAT.H, ;WHEN THE R80 FORMAT BIT SETS..BRANCH
:22730 NAD/GROUP0 ; TO NEXT WORD AND THEN BRANCH ON THE
:22731 ; FUNCTION BITS
:22732 -----
:22733 167:
:22734 =1**
:22735
:22736 BEN0/F0,
:22737 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:22738 BEN2/F2,
:22739 NAD/GROUP0.F0
:22740 -----
:22741 021:

```

[illegible]

U 14  
U 14  
U 14  
  
U 14  
U 14  
  
U 14  
UU 14  
UU 14  
UU 14  
UU 14  
UU 14  
UU 14

```

:22838 .PAGE "Test 29 DC007 INTERNAL SHIFT REGISTER INIT TEST (M8388 IDC MODULE)"
:22839 .++
:22840
:22841 .Test Description:
:22842
:22843 .This test is designed to insure that none of the bits of the internal
:22844 .shift register of the CRC/ECC chip are stuck at one. This will be done
:22845 .by asserting UCLR ECC which should clear the shift register. The CPU
:22846 .will then do a READ PATTERN, which reads bits <31:21> of the shift
:22847 .register and check for all zeros. The register will then be shifted
:22848 .left B Hex times while shifting in zeros. The CPU will then do a
:22849 .READ PATTERN and check for all zeros. The register will again be
:22850 .shifted left B Hex times and the CPU will READ PATTERN and check for
:22851 .all zeros.
:22852
:22853 .Assumptions:
:22854
:22855 .It is assumed that all previous test have run successfully.
:22856
:22857 .Test Steps:
:22858
:22859 .1. Set up error mask, error number and module number in LS
:22860 .(for error printouts) and clear the previous error number in LS.
:22861 .2. Force the IDC microcode to DIAG.IDLE.
:22862 .3. Set up the CSR to branch to the correct IDC microcode routine.
:22863
:22864 .*****
:22865 .* IDC *
:22866 .*****
:22867
:22868 .IDC1. Execute an instruction with FUNC <35:32> = 0001,
:22869 .UCLR ECC.
:22870 .IDC2. Send XFER REQ.
:22871
:22872 .4. Wait for an XFER REQ from IDC.
:22873 .5. Clear CSR F<2:0> and send XFER GRANT.
:22874 .6. Execute a READ PATTERN and check the data = 0.
:22875 .7. Increment the error number.
:22876 .8. Set up the CSR to branch to the correct IDC routine.
:22877
:22878
:22879 .*****
:22880 .* IDC *
:22881 .*****
:22882
:22883 .IDC3. Shift the pattern register B HEX times.
:22884 .IDC4. Send XFER REQ.
:22885
:22886 .9. Wait for XFER REQ from the IDC.
:22887 .10. Clear CSR F<2:0> and send XFER GRANT.
:22888 .11. Read PATTERN and check data = 0.
:22889 .12. Increment the error number.
:22890 .13. Repeat Steps 8 - 11 to read the remaining bits of the register.
:22891 .14. Clear IDC.
:22892 .15. End test.

```



ZZ-E  
:  
:  
  
U 14  
U 14  
U 14  
  
  
  
  
  
  
  
  
  
U 14  
U 14  
U 14  
U 14  
U 14  
U 15  
U 15  
U 15  
  
  
  
U 15

```
:22893 :
:22894 : Errors:
:22895 :
:22896 : Printout:
:22897 : EXPECTED RECEIVED
:22898 : INTERNAL SHIFT REG INTERNAL SHIFT REG
:22899 : (ECC PATTERN) (ECC PATTERN)
:22900 :
:22901 : ERROR 1 - The CRC/ECC internal shift register failed or the READ
:22902 : PATTERN instruction failed.
:22903 :
:22904 : ERROR 2 - The CRC/ECC internal shift register failed.
:22905 :
:22906 : ERROR 3 - The CRC/ECC internal shift register failed.
:22907 :
:22908 :
:22909 : Debug:
:22910 :
:22911 : ERROR 1 -
:22912 : The IDC microcode will execute an instrcution with FUNC <3:0>=0001.
:22913 : These bits will be decoded and will assert UCLR ECC L. This signal
:22914 : is the Clear line to the CRC/ECC chip and will clear the CRC/ECC
:22915 : chip internal Pattern register.
:22916 : The CPU will issue a MISC [SET.PORT.I.0] instruction that will
:22917 : deassert SEL ACC IN H. This is inverted to SEL ACC L = H.
:22918 : The CPU will then issue a PORT.READ PORT.ADRS[PATTERN] instruction.
:22919 : The inputs to the PORT INTERFACE REG PAL will be:
:22920 : CSR 17 H = H
:22921 : CSR 14 H = L
:22922 : CSR 12 H = H
:22923 : CSR 11 H = L
:22924 : CSR 10 H = L
:22925 : PORT INSTR H = H
:22926 : CPU P2 H = H
:22927 : At CPU CLOCK H, the outputs will be:
:22928 : SEL IDC H = H
:22929 : R2 L = L
:22930 : R1 L = H
:22931 : R0 L = H
:22932 : The next instruction issued by the CPU will be a MOV PORT TO LS[].
:22933 : This will assert READ PORT L. This signal is inverted by the PORT
:22934 : INTERFACE REG PAL and becomes READ PORT H = H. READ PORT H = H
:22935 : is gated with SEL IDC H = H and SEL ACC L = H and will assert READ
:22936 : IDC L. READ IDC L = L and READ PORT H = H are the enables for the
:22937 : decoder of R2 L = L, R1 L = H, R0 L = H. This combination of inputs
:22938 : will assert SEL PATTERN L. SEL PATTERN L is an input to the CRC/ECC
:22939 : chip and will drive the internal Pattern register bits<21:31> onto
:22940 : BUS IO <10:0>. This data is then stored in the specified LS location.
:22941 :
:22942 : ERROR 2 -
:22943 : The CRC/ECC internal Pattern register is shifted 11 times so that
:22944 : the next 11 bits can be read.
:22945 : This is done by executing 11 IDC microinstructions with UENB CRC H = L,
:22946 : UDATA FIELD H = H, UMAINT H = H. UMAINT H = H allows DSR 0 to become
:22947 : NRZ WRT DATA H at the MFM ENCODER PAL so that zeros continue to
```

```

ZZ-ENKCF-1.4 ; ENKCF.MIC Test 29 DC007 INTE15-MAR-83 J 5 Fiche 3 Frame J5 Sequence 473 Page 467 ZZ-EI
: ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40
: ENKCF.MIC Test 29 DC007 INTERNAL SHIFT REGISTER INIT TEST (M8388 IDC MODULE)

:22948 : be shifted into the CRC/ECC internal Pattern register. UDATA FIELD H
:22949 : allows the Pattern register to be clocked and UENB CRC H = L allows
:22950 : all 32 bits of the internal Pattern register to shift. The CPU will
:22951 : then read bits<21:31> of the Pattern register.
:22952 :
:22953 : ERROR 3 -
:22954 : The CRC/ECC internal Pattern register is shifted 11 times so that
:22955 : the last 11 bits can be read.
:22956 : See ERROR 2.
:22957 : --
:22958 : *****
:22959 : T.29:
U 147D, B65E,15 :22960 : MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT 15 IN WR[0] FOR
:22961 : ; CONTROL AND STATUS WORD
U 147E, 3E80,15 :22962 : MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND
:22963 : ; STATUS WORD - BIT 15
:22964 : ; INDICATES START OF TEST TO
:22965 : ; CONSOLE
U 147F, 10E0,15 :22966 : MISC [SET.CP.ATTN] ;ISSUE CPU ATTN TO CONSOLE
:22967 : WAIT.T29.0:
U 1480, 0948,04 :22968 : JMP [WAIT.T29.0] ;LOOP HERE WAITING FOR
:22969 : ; CONSOLE TO RESPOND
:22970 :
U 1481, 65A0,15 :22971 : CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
:22972 : ; IN LS
:22973 :
U 1482, B64A,15 :22974 : MOV LS[IDC] TO WR[0] ;STORE MODULE CODE IN LS TO
U 1483, 3E8C,15 :22975 : MOV WR[0] TO LS[MODULE.NUM] ; INDICATE IDC MODULE
:22976 :
:22977 :
:22978 :
U 1484, B7A6,15 :22979 : MOV LS[#FFFFE000] TO WR[0] ;SET UP THE MASK TO CHECK
U 1485, 3E8A,15 :22980 : MOV WR[0] TO LS[ERROR.MASK] ; BITS <12:0>
:22981 :
U 1486, 8872,EC :22982 : JSR [DIAG.CODE] ;FORCE THE IDC TO DIAG.IDLE
:22983 :
:22984 : LOOP.T29.1:
U 1487, B640,15 :22985 : MOV LS[#1] TO WR[0] ;SET WRO = 1
U 1488, BE82,15 :22986 : MOV WR[0] TO LS[ERROR.NUMBER] ;SET UP ERROR NUMBER = 1 IN LS
:22987 :
U 1489, B712,15 :22988 : MOV LS[SETCSR.CRDY] TO WR[0] ;SET UP THE CSR TO BRANCH TO
U 148A, B720,95 :22989 : MOV LS[SETCSR.MAINT] TO WR[1] ; THE CORRECT IDC ROUTINE
U 148B, AEC2,15 :22990 : BIS WR[1] TO WR[0]
U 148C, 370C,95 :22991 : MOV LS[SETCSR.F6] TO WR[1]
U 148D, AEC2,15 :22992 : BIS WR[1] TO WR[0]
U 148E, 17A0,15 :22993 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:22994 : ;MAINT=1,CRDY=1,F<2:0>=110
:22995 : UCLR ECC
:22996 : -----
:22997 : -----
:22998 : *****
:22999 : * IDC *
:3000 : *****
:3001 : -----
:3002 : 018:

```

```

:23003 :=000
:23004 :DIAG.IDLE:
:23005 :
:23006 : BEN0/F0,
:23007 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:23008 : BEN2/F2,
:23009 : NAD/DIAG.IDLE
:23010 :-----
:23011 :01E:
:23012 :=110
:23013 :
:23014 : BEN0/CRDY.L,
:23015 : BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:23016 : NAD/TEST.FUNC6 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:23017 :-----
:23018 :076:
:23019 :=1*0
:23020 :TEST.ECC.1:
:23021 :
:23022 : FUNC/CLR.ECC, ;ISSUE CLR ECC TO INIT THE DC007
:23023 : NAD/XFER.REQ ;FLAG THE CPU
:23024 :-----
:23025 :1D8:
:23026 :XFER.REQ:
:23027 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:23028 : NAD/DIAG.WAIT
:23029 :-----
:23030 :1D7:
:23031 :DIAG.WAIT:
:23032 :
:23033 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:23034 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:23035 :-----
:23036 :159:
:23037 :=*0*
:23038 :DIAG.WAIT1:
:23039 :
:23040 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:23041 : ; COMMAND FROM THE CPU
:23042 :-----
:23043 :15B:
:23044 :=*1*
:23045 :DIAG.WAIT2:
:23046 :
:23047 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:23048 : ; WAIT SOME MORE
:23049 :-----
:23050 :WAIT.IDC.T29.1:
:23051 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ FROM IDC
:23052 : JMP [IDC.DONE.T29.1] ;XFER REQ SET
:23053 : JMP [WAIT.IDC.T29.1] ;WAIT SOME MORE
:23054 :
:23055 :IDC.DONE.T29.1:
:23056 : MOV LS[SETCSR.F0] TO WR[0] ;CLEAR CSR F<2:0>
:23057 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR

```

U 148F, DB00,1A  
 U 1490, 0949,24  
 U 1491, 0948,F4

U 1492, B700,15  
 U 1493, 17A0,15

013

ZZ-ENKCF-1.4 :  
: ENKCF.MCR  
: ENKCF.MIC

ENKCF.MIC

MICRO2 1M(01)

Test 29 DC007 INTE15-MAR-83

15-MAR-83 11:46:22

ENKCF Micro-diagnostic for IDC module

Sequence 476

Page 470

Test 29 DC007 INTERNAL SHIFT REGISTER INIT TEST (M8388 IDC MODULE)

ZZ-EN

:

:

U 150

U 150

U 150

U 150

U 151

U 151

U 151

U 151

U 151

U 151

U 151

```
:23113 :078:
:23114 :=0*0
:23115 :TEST.FUNC7:
:23116 :
:23117 : NAD/GROUP0
:23118 :-----
:23119 :163:
:23120 :=0**
:23121 :GROUP0:
:23122 :
:23123 : BEN2/R80.FORMAT.H, ;WHEN THE R80 FORMAT BIT SETS..BRANCH
:23124 : NAD/GROUP0 ; TO NEXT WORD AND THEN BRANCH ON THE
:23125 : ; FUNCTION BITS
:23126 :-----
:23127 :167:
:23128 :=1**
:23129 :
:23130 : BEN0/F0,
:23131 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:23132 : BEN2/F2,
:23133 : NAD/GROUP0.F0
:23134 :-----
:23135 :023:
:23136 :=011
:23137 :DC007.SHIFT.REG.1:
:23138 :
:23139 : LOAD.LOOP.CNTR.[OF5], ;SHIFT THE DC007 INT REG 11 TIMES
:23140 : NAD/DC007.SHIFT.REG.2
:23141 :-----
:23142 :194:
:23143 :=0
:23144 :DC007.SHIFT.REG.2:
:23145 :
:23146 : MODE/ECC, ;ASSERT ECC TO SHFT BIT 31
:23147 : DATA.FIELD/ON, ;TO ALLOW READ CLK
:23148 : UMAINT/ON, ;TO SELECT DSRO AS NRZ DATA
:23149 : INCR.LPCNTR/ON,
:23150 : BEN0/CNTR.OVFLW, ;LOOP UNTIL SHIFT 11 TIMES
:23151 : NAD/DC007.SHIFT.REG.2
:23152 :-----
:23153 :195:
:23154 :=1
:23155 :DC007.SHIFT.REG.3:
:23156 :
:23157 : NAD/XFER.REQ ;FLAG THE CPU
:23158 :-----
:23159 :1D8:
:23160 :XFER.REQ:
:23161 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:23162 : NAD/DIAG.WAIT
:23163 :-----
:23164 :1D7:
:23165 :DIAG.WAIT:
:23166 :
:23167 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
```

|   |     |
|---|-----|
| U | 151 |
| U | 151 |
| U | 151 |
|   |     |
| U | 151 |
| U | 151 |
| U | 151 |
|   |     |
| U | 151 |
| U | 151 |
| U | 151 |
|   |     |
| U | 152 |
| U | 152 |
| U | 152 |
| U | 152 |
|   |     |
| U | 152 |
| U | 152 |
|   |     |
| U | 152 |
| U | 152 |
| U | 152 |
| U | 152 |
| U | 152 |
|   |     |
| U | 152 |
| U | 152 |
| U | 152 |
| U | 152 |

:23214 :PAGE 'Test 2A DC007 PATTERN REGISTER TEST (M8388 IDC MODULE)'  
:23215 :++  
:23216 :

:23217 :Test Description:  
:23218 :  
:23219 :

:23220 :This test is designed to insure that the internal shift register  
:23221 : (pattern register) of the DC007 chip is working correctly.  
:23222 : This will be accomplished by first clearing the register. A one is then  
:23223 : shifted from the DSR and into bit 0 of the pattern register. The  
:23224 : pattern register is then read and checked. The pattern register is then  
:23225 : shifted one time and read and checked again until it has been shifted  
:23226 : 105 times. The pattern register does not perform a normal shift  
:23227 : function. When in ECC mode, Bits<23,21,11,2> become the XOR of Feedback  
:23228 : and the preceding bit. In this test Feedback = Bit 31.  
:23229 :

:23230 :Assumptions:  
:23231 :

:23232 :It is assumed that all previous tests have run successfully.  
:23233 :

:23234 :Test Steps:  
:23235 :

- :23236 :1. Set up error mask, error number and module number in LS  
:23237 : (for error printouts) and clear the previous error number in LS.  
:23238 :2. Force the IDC microcode to DIAG.IDLE.  
:23239 :3. Write a one to FIFO A.  
:23240 :4. Set up the CSR to branch to the correct IDC microcode routine.

:23241 :\*\*\*\*\*  
:23242 : \* IDC \*  
:23243 :\*\*\*\*\*

- :23244 :IDC1. Execute an instruction with FUNC <35:32> = 0001,  
:23245 : UCLR ECC.  
:23246 :IDC2. Load the data pattern from the FIFO and into the Data  
:23247 : Shift Register.  
:23248 :IDC3. Assert UMAINT to allow DSRO to become SDIO.(NRZ DATA).  
:23249 :IDC4. Shift the one into the DC007 internal shift register.  
:23250 :IDC5. Send XFER REQ to flag the CPU.

- :23251 :5. Wait for XFER REQ from the IDC.  
:23252 :6. Clear CSR F<2:0> and send XFER GRANT.  
:23253 :7. Check the pattern register.  
:23254 :8. Increment the error number.  
:23255 :9. Set up the CSR to branch to the correct IDC routine.

:23256 :\*\*\*\*\*  
:23257 : \* IDC \*  
:23258 :\*\*\*\*\*

- :23259 :IDC5. Allow the pattern register to clock once.  
:23260 :IDC6. Send XFER REQ to flag the CPU.

- :23261 :10. Wait for XFER REQ from the IDC.  
:23262 :11. Clear CSR F<2:0> and send XFER GRANT.  
:23263 :12. Check the pattern register.  
:23264 :  
:23265 :  
:23266 :  
:23267 :  
:23268 :

:23269 : 13. Repeat Steps 8 - 13 105 times.  
:23270 : 14. Clear IDC.  
:23271 : 15. End test.

:23272 :  
:23273 : Errors:

:23274 :  
:23275 : Printout:  
:23276 : EXPECTED RECEIVED  
:23277 : INTERNAL SHIFT REG INTERNAL SHIFT REG  
:23278 : (ECC PATTERN) (ECC PATTERN)  
:23279 :

:23280 : ERROR 1 - CRC/ECC chip internal shift register failed when a one was  
:23281 : shifted into the Pattern register.

:23282 :  
:23283 : ERROR 2 - CRC/ECC chip internal shift register failed when the data  
:23284 : was being shifted through the Pattern register.  
:23285 :

:23286 : Debug:

:23287 :  
:23288 : ERROR 1 -

:23289 : A data pattern of one is written to FIFO A address 0.  
:23290 : The IDC microcode will execute and instruction with F<2:0>=0001 which  
:23291 : will be decoded and assert UCLR ECC L. This signal will clear the  
:23292 : CRC/ECC internal Pattern register.  
:23293 : The data pattern is loaded into the DSR and UMAINT H is asserted which  
:23294 : allows DSR 0 H to become NRZ WRT DATA H. In the next IDC  
:23295 : micro-instruction, UDATA FIELD H is asserted and UENB CRC H is  
:23296 : deasserted. UDATA FIELD H = H allows NRZ WRT DATA H to be shifted into  
:23297 : the CRC/ECC Pattern register and UENB CRC H = L enables some internal  
:23298 : signals of the CRC/ECC chip. The CPU will then read bits <21:31> of the  
:23299 : Pattern register on BUS IO <10:0>. The CRC/ECC chip internal Pattern  
:23300 : register does NOT perform a normal shift function. There is some  
:23301 : XORing of certain signals. The CPU microcode will calculate the  
:23302 : expected data.  
:23303 :

:23304 : ERROR 2 -

:23305 : The data in the CRC/ECC chip internal Pattern register will be shifted  
:23306 : once, and then the CPU microcode will read bits <21:31> of the Pattern  
:23307 : register on BUS IO <10:0>. This will be done 105 (decimal) times to  
:23308 : fully test the CRC/ECC internal Pattern register.  
:23309 : To shift the CRC/ECC chip internal Pattern register, the IDC will  
:23310 : execute an instruction with UDATA FIELD H = H, UMAINT H = H and  
:23311 : UENB CRC H = L. UDATA FIELD H = H enables the internal clocks.  
:23312 : UENB CRC H = L allows certain internal signals. UMAINT H = H  
:23313 : allows zeros to be shifted in as NRZ WRT DATA H.  
:23314 :

:23315 : --

:23316 : \*\*\*\*\*

:23317 : T.2A:

U 14B9, B65E,15

:23318 : MOV LS[BEGIN.TEST] TO WR[0]

:23319 :

U 14BA, 3E80,15

:23320 : MOV WR[0] TO LS[CONTROL.STATUS]

:23321 :

:23322 :

:23323 :

: SET BIT 15 IN WR[0] FOR  
: CONTROL AND STATUS WORD  
: SET BIT 15 IN CONTROL AND  
: STATUS WORD - BIT 15  
: INDICATES START OF TEST TO  
: CONSOLE



```

U 14BB, 10E0,15 :23324 MISC [SET.CP.ATTN] ;ISSUE CPU ATTN TO CONSOLE
:23325 WAIT.T2A.0:
U 14BC, 094B,C4 :23326 JMP [WAIT.T2A.0] ;LOOP HERE WAITING FOR
:23327 ; CONSOLE TO RESPOND
:23328
U 14BD, 65A0,15 :23329 CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
:23330 ; IN LS
:23331
U 14BE, B64A,15 :23332 MOV LS[IDC] TO WR[0] ;STORE MODULE CODE IN LS TO
U 14BF, 3E8C,15 :23333 MOV WR[0] TO LS[MODULE.NUM] ; INDICATE IDC MODULE
:23334
:23335
:23336
U 14C0, 37AE,15 :23337 MOV LS[FFFFFF800] TO WR[0] ;SET UP THE MASK TO CHECK
U 14C1, 3E8A,15 :23338 MOV WR[0] TO LS[ERROR.MASK] ; BITS <10:0>
:23339
:23340
U 14C2, 8872,EC :23341 JSR [DIAG.CODE] ;FORCE THE IDC TO DIAG.IDLE
:23342
:23343 LOOP.T2A.2:
U 14C3, 37A3,95 :23344 MOV LS[A0100500] TO WR[3] ;EXPECTED DATA
U 14C4, 17D8,15 :23345 PORT.CONTROL [SEL.FIFO.A] ;SELECT FIFO A
U 14C5, 17C0,15 :23346 PORT.CONTROL [CLEAR.FIFO.CNTR] ;SET FIFO A ADRS = 0
U 14C6, B640,15 :23347 MOV LS[#1] TO WR[0] ;GET THE DATA TO WRITE
U 14C7, 97A8,15 :23348 PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;WRITE A ONE TO FIFO A
:23349
:23350 LOOP.T2A.1:
U 14C8, B640,15 :23351 MOV LS[#1] TO WR[0] ;SET WRO = 1
U 14C9, BE82,15 :23352 MOV WR[0] TO LS[ERROR.NUMBER] ;SET UP ERROR NUMBER = 1 IN LS
:23353
:23354 PORT.CONTROL [CLEAR.FIFO.CNTR] ;SET FIFO ADRS =0
U 14CA, 17C0,15 :23355 MOV LS[SETCSR.CRDY] TO WR[1] ;SET UP THE CSR TO BRANCH TO
U 14CB, 3712,95 :23356 MOV LS[SETCSR.F7] TO WR[0] ; THE CORRECT IDC ROUTINE
U 14CC, 370E,15 :23357 BIS WR[1] TO WR[0]
U 14CD, AEC2,15 :23358 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
U 14CE, 17A0,15 :23359 ;MAINT=0,CRDY=1,F<2:0>=111
:23360
:23361 ;CONTINUE BRANCHING
U 14CF, 379E,15 :23362 MOV LS[SETCSR.R80] TO WR[0] ;CONTINUE TO BRANCH TO ROUTINE
U 14D0, 3700,95 :23363 MOV LS[SETCSR.F0] TO WR[1]
U 14D1, 3713,15 :23364 MOV LS[SETCSR.CRDY] TO WR[2] ;SET CRDY TO HOLD SDIO LOW
U 14D2, AEC4,15 :23365 BIS WR[2] TO WR[0]
U 14D3, AEC2,15 :23366 BIS WR[1] TO WR[0]
U 14D4, 17A0,15 :23367 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:23368 ;R80=1, F<2:0>=000
:23369 -----
:23370 *****
:23371 * IDC *
:23372 *****
:23373 -----
:23374 :018:
:23375 :=000
:23376 :DIAG.IDLE:
:23377 :
:23378 : BEN0/F0,

```

|   |   |   |   |   |   |
|---|---|---|---|---|---|
| U | 1 | 1 | 1 | 1 | 1 |
| U | U | U | U | U | U |

```

23434 :-----
23435 :1E8:
23436 :PAT.REG.4:
23437 :
23438 : DATA.FIELD/ON, ;ALLOW READ CLK
23439 : MODE/ECC, ;ECC MODE TO SHIFT BIT 31
23440 : NAD/XFER.REQ ;FLAG THE CPU
23441 :-----
23442 :1D8:
23443 :XFER.REQ:
23444 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
23445 : NAD/DIAG.WAIT
23446 :-----
23447 :1D7:
23448 :DIAG.WAIT:
23449 :
23450 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
23451 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
23452 :-----
23453 :159:
23454 :=*0*
23455 :DIAG.WAIT1:
23456 :
23457 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
23458 : ; COMMAND FROM THE CPU
23459 :-----
23460 :15B:
23461 :=*1*
23462 :DIAG.WAIT2:
23463 :
23464 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
23465 : ; WAIT SOME MORE
23466 :-----
23467 :WAIT.IDC.T2A.1:
23468 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ FROM IDC
23469 : JMP [IDC.DONE.T2A.1] ;XFER REQ SET
23470 : JMP [WAIT.IDC.T2A.1] ;WAIT SOME MORE
23471 :
23472 :IDC.DONE.T2A.1:
23473 : MOV LS[SETCSR.F0] TO WR[0] ;CLEAR CSR F<2:0>
23474 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
23475 : MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
23476 :
23477 :READ.PAT.T2A.1:
23478 : MISC [SET.PORT.I.0] ;SELECT THE IDC
23479 : PORT.READ PORT.ADRS[PATTERN] ;SET UP TO READ PATTERN REG
23480 : MOV PORT TO LS[PORT0] ;READ THE DATA FROM PAT AND
23481 : ; SAVE IT IN LS
23482 : MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
23483 : MOV WR[3] TO WR[1] ;SET UP EXPECTED DATA
23484 : JSR [CHECK.RESULT] ;CHECK THE DATA
23485 : JMP [LOOP.T2A.1] ;LOOP HERE IF ERR & LOE IS SET
23486 :
23487 : INC LS[ERROR.NUMBER] ;ERROR 2
23488 : MOV LS[#69] TO WR[2] ;USE AS A LOOP COUNTER=105

```

U 14D5, DB00,1A  
U 14D6, 894D,84  
U 14D7, 094D,54

U 14D8, B700,15  
U 14D9, 17A0,15  
U 14DA, 90FA,15

U 14DB, 9090,15  
U 14DC, 1790,15  
U 14DD, 3B32,15

U 14DE, 3732,15  
U 14DF, 2006,95  
U 14E0, 0869,3C  
U 14E1, 094C,84

U 14E2, FF82,15  
U 14E3, B7A9,15

U 1  
U 1  
U 1

U 1  
U 1  
U 1

U 1  
U 1  
U 1

U 1  
U 1  
U 1  
U 1  
U 1

```

:23489
:23490 CHK.PAT.REG.T2A:
:23491
:23492 ;SET UP THE NEW EXPECTED DATA
U 14E4, 2341,95 :23493 ROR WR[3] ;ROTATE THE 32 BITS OF DATA
:23494 BIT LS[BIT31] WITH WR[3], ;IS BIT 31 SET?
U 14E5, D97F,B5 :23495 DT(LONG)&SET.ALU.CC
U 14E6, 894E,99 :23496 JMP.IF[BITS.CLR] TO [NO.XOR.T2A.1] ;JMP IF BIT31 = 0 WR[3] HAS
:23497 ; EXPECTED DATA
U 14E7, B7A0,15 :23498 MOV LS[#20100500] TO WR[0] ;GET THE MASK
U 14E8, 2F81,95 :23499 XOR WR[0] TO WR[3] ;GET THE NEW EXPECTED DATA
:23500
:23501 NO.XOR.T2A.1:
U 14E9, 3712,95 :23502 MOV LS[SETCSR.CRDY] TO WR[1] ;SET UP THE CSR TO BRANCH TO
U 14EA, 370E,15 :23503 MOV LS[SETCSR.F7] TO WR[0] ; THE CORRECT IDC ROUTINE
U 14EB, AEC2,15 :23504 BIS WR[1] TO WR[0]
U 14EC, 17A0,15 :23505 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:23506 ;MAINT=0,CRDY=1,F<2:0>=111
:23507 ;CONTINUE BRANCHING
U 14ED, 379E,15 :23508 MOV LS[SETCSR.R80] TO WR[0] ;CONTINUE TO BRANCH
U 14EE, B708,95 :23509 MOV LS[SETCSR.F4] TO WR[1]
U 14EF, AEC2,15 :23510 BIS WR[1] TO WR[0]
U 14F0, 3712,95 :23511 MOV LS[SETCSR.CRDY] TO WR[1] ;SET CRDY TO HOLD SDIO =LO
U 14F1, AEC2,15 :23512 BIS WR[1] TO WR[0]
U 14F2, 17A0,15 :23513 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:23514 ;R80=1,F<2:0>=100
:23515 -----
:23516 -----
:23517 *****
:23518 * IDC *
:23519 *****
:23520 -----
:23521 018:
:23522 =000
:23523 DIAG.IDLE:
:23524
:23525 BEN0/F0,
:23526 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:23527 BEN2/F2,
:23528 NAD/DIAG.IDLE
:23529 -----
:23530 01F:
:23531 =111
:23532
:23533 BEN0/CRDY.L,
:23534 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:23535 NAD/TEST.FUNC7 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:23536 -----
:23537 078:
:23538 =0*0
:23539 TEST.FUNC7:
:23540
:23541 NAD/GROUP0
:23542 -----
:23543 163:

```

```

:23544 :=0**
:23545 :GROUP0:
:23546 :
:23547 : BEN2/R80.FORMAT.H, ;WHEN THE R80 FORMAT BIT SETS..BRANCH
:23548 : NAD/GROUP0 ; TO NEXT WORD AND THEN BRANCH ON THE
:23549 : ; FUNCTION BITS
:23550 :-----
:23551 :167:
:23552 :=1**
:23553 :
:23554 : BEN0/F0,
:23555 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:23556 : BEN2/F2,
:23557 : NAD/GROUP0.F0
:23558 :-----
:23559 :024:
:23560 :=100
:23561 :SHFT.PAT.REG.1:
:23562 :
:23563 : MODE/ECC, ;SET MODE ECC TO ALLOW BIT 31 TO SHIFT
:23564 : DATA.FIELD/ON, ;ALLOW READ CLK
:23565 : UMAINT/ON, ;TO SELECT DSRO AS NRZ DATA
:23566 : NAD/XFER.REQ ;FLAG THE CPU
:23567 :-----
:23568 :1D8:
:23569 :XFER.REQ:
:23570 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:23571 : NAD/DIAG.WAIT
:23572 :-----
:23573 :1D7:
:23574 :DIAG.WAIT:
:23575 :
:23576 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:23577 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:23578 :-----
:23579 :159:
:23580 :=*0*
:23581 :DIAG.WAIT1:
:23582 :
:23583 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:23584 : ; COMMAND FROM THE CPU
:23585 :-----
:23586 :15B:
:23587 :=*1*
:23588 :DIAG.WAIT2:
:23589 :
:23590 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:23591 : ; WAIT SOME MORE
:23592 :-----
:23593 :WAIT.IDC.T2A.2:
:23594 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ FROM IDC
:23595 : JMP [IDC.DONE.T2A.2] ;XFER REQ SET
:23596 : JMP [WAIT.IDC.T2A.2] ;WAIT SOME MORE
:23597 :
:23598 :IDC.DCNE.T2A.2:

```

U 14F3, DB00,1A  
 U 14F4, 894F,64  
 U 14F5, 894F,34

```

U 14F6, B700,15 :23599 MOV LS[SETCSR.F0] TO WR[0] ;CLEAR CSR F<2:0>
U 14F7, 17A0,15 :23600 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
U 14F8, 90FA,15 :23601 MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
:23602
:23603
:23604 READ.PAT.T2A.2:
U 14F9, 9090,15 :23605 MISC [SET.PORT.I.0] ;SELECT THE IDC
U 14FA, 1790,15 :23606 PORT.READ PORT.ADRS[PATTERN] ;SET UP TO READ PATTERN REG
U 14FB, 3B32,15 :23607 MOV PORT TO LS[PORT0] ;READ THE DATA FROM PAT AND
:23608 ; SAVE IT IN LS
U 14FC, 3732,15 :23609 MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
U 14FD, 2006,95 :23610 MOV WR[3] TO WR[1] ;SET UP THE EXPECTED DATA
U 14FE, 0869,3C :23611 JSR [CHECK.RESULT] ;CHECK THE DATA
U 14FF, 894C,34 :23612 JMP [LOOP.T2A.2] ;LOOP HERE IF ERR & LOE IS SET
:23613 DEC WR[2], ;DONE?
U 1500, A105,35 :23614 DT(LONG)&SET.ALU.CC
U 1501, 894E,41 :23615 JMP.IF[NEQ] TO [CHK.PAT.REG.T2A] ;JMP IF NOT DONE
U 1502, 0950,34 :23616 JMP [CLEANUP.T2A]
:23617
:23618
:23619 CLEANUP.T2A:
U 1503, 8875,3C :23620 JSR [CLEANUP] ;CLEANUP THE CONDITIONS SET BY
:23621 ; THIS TEST
:23622 END.T2A:
:23623
:23624

```

22-E  
:  
:

U 15

U 15  
U 15  
U 15  
U 15

U 15  
U 14  
U 14

U 19  
U 19  
U 19  
U 19  
U 19

:23625 :PAGE "Test 2B DC007 POSITION REGISTER TEST(M8388 IDC MODULE)"  
:23626 :++  
:23627 :  
:23628 :Test Description:  
:23629 :  
:23630 :This test is designed to verify that the position register of the  
:23631 :DC007 chip is working correctly. A UCLR ECC will be executed and the  
:23632 :position register will be checked for the initial value of 06837 Hex  
:23633 : (064067 octal). The register will then be clocked on tick at a time  
:23634 : and read to insure that the register incremented. When the limit value  
:23635 : 1021 HEX is reached the register will be clocked one more time and the  
:23636 : data should remain the same.  
:23637 :  
:23638 :Assumptions:  
:23639 :  
:23640 :It is assumed that all previous tests have run successfully.  
:23641 :  
:23642 :Test Steps:  
:23643 :  
:23644 :1. Set up error mask, error number and module number in LS  
:23645 : (for error printouts) and clear the previous error number in LS.  
:23646 :2. Force the IDC microcode to DIAG.IDLE.  
:23647 :3. Set up the CSR to branch to the correct IDC microcode routine.  
:23648 :  
:23649 :  
:23650 :\*\*\*\*\*  
:23651 : \* IDC \*  
:23652 :\*\*\*\*\*  
:23653 :  
:23654 :IDC1. Execute an instruction with FUNC <35:32> = 0001,  
:23655 : UCLR ECC.  
:23656 :  
:23657 :4. Wait for a XFER REQ from the IDC.  
:23658 :5. Clear CSR F<2:0> and send XFER GRANT.  
:23659 :6. READ POSITION and check for lower 13 bits of 06837 Hex  
:23660 : (064067 octal).  
:23661 :7. Set up the CSR to branch to the correct IDC routine.  
:23662 :  
:23663 :  
:23664 :\*\*\*\*\*  
:23665 : \* IDC \*  
:23666 :\*\*\*\*\*  
:23667 :  
:23668 :IDC2. Assert Error to allow the position register to clock.  
:23669 :IDC3. Clock the position register.  
:23670 :IDC4. Send XFER REQ to flag the CPU.  
:23671 :  
:23672 :8. Wait for an XFER REQ from the IDC.  
:23673 :9. Clear CSR F<2:0> and send XFER GRANT.  
:23674 :10. Check the Position Register data.  
:23675 :11. Set up the CSR to branch to the correct IDC routine.  
:23676 :  
:23677 :\*\*\*\*\*  
:23678 : \* IDC \*  
:23679 :\*\*\*\*\*

```

:23680 :
:23681 : IDC5. Clock the position register.
:23682 : IDC6. Send XFER REQ to flag the CPU.
:23683 :
:23684 : 12. Wait for an XFER REQ from the IDC.
:23685 : 13. Clear CSR F<2:0> and send XFER GRANT.
:23686 : 14. Check the Position Register.
:23687 : 15. Repeat Steps 11 - 15 until Limit (1021 HEX).
:23688 : 16. Repeat Steps 11 - 15 and check that the Position Register
:23689 : data did not change.
:23690 : 17. Clear IDC.
:23691 : 18. End test.

```

:23692 : Errors:

```

:23693 : Printout:
:23694 :
:23695 : EXPECTED RECEIVED
:23696 : ECC POSITION ECC POSITION
:23697 :
:23698 :
:23699 : ERROR 1 - The CRC/ECC chip internal position register failed or
:23700 : the READ POSITION instruction failed.
:23701 :
:23702 : ERROR 2 - The CRC/ECC chip internal Position register failed.
:23703 :
:23704 : ERROR 3 - The CRC/ECC chip internal Position register failed when
:23705 : LIMIT was reached.
:23706 :
:23707 : ERROR 4 - CSR ECC STAT<1:0> failed or CRC/ECC failed.

```

:23708 : Debug:

```

:23709 :
:23710 : ERROR 1 -
:23711 :
:23712 : The IDC microcode will execute an instruction with FUNC <3:0>=0001.
:23713 : These bits will be decoded and will assert UCLR ECC L. This signal
:23714 : is the Clear line to the CRC/ECC chip and will initialize the CRC/ECC
:23715 : chip internal Position register to 6837 HEX. Only bits <12:0> can be
:23716 : read by the CPU.
:23717 : The CPU will issue a MISC [SET.PORT.I.O] instruction that will
:23718 : deassert SEL ACC IN H. This is inverted to SEL ACC L = H.
:23719 : The CPU will then issue a PORT.READ PORT.ADRS[POSITION] instruction.
:23720 : The inputs to the PORT INTERFACE REG PAL will be:
:23721 :
:23722 : CSR 17 H = H
:23723 : CSR 14 H = L
:23724 : CSR 12 H = H
:23725 : CSR 11 H = L
:23726 : CSR 10 H = H
:23727 : PORT INSTR H = H
:23728 : CPU P2 H = H
:23729 : At CPU CLOCK H, the outputs will be:
:23730 :
:23731 : SEL IDC H = H
:23732 : R2 L = L
:23733 : R1 L = H
:23734 : R0 L = L

```

The next instruction issued by the CPU will be a MOV PORT TO LS[ ].



[illegible]

|              |   |           |                                                        |                                       |              |      |
|--------------|---|-----------|--------------------------------------------------------|---------------------------------------|--------------|------|
| ZZ-ENKCF-1.4 | : | ENKCF.MIC | Test 2B DC007 POS15-MAR-83                             | Fiche 3 Frame M6                      | Sequence 489 | ZZ-E |
| :            | : | ENKCF.MCR | MICRO2 1M(01) 15-MAR-83 11:46:22                       | ENKCF Micro-diagnostic for IDC module | Rev 01.40    | :    |
| :            | : | ENKCF.MIC | Test 2B DC007 POSITION REGISTER TEST(M8388 IDC MODULE) |                                       | Page 483     | :    |

  

|                 |   |       |                                      |                                              |      |
|-----------------|---|-------|--------------------------------------|----------------------------------------------|------|
| U 150C, 3E8A,15 | : | 23790 | MOV WR[0] TO LS[ERROR.MASK]          | ; CHECK BITS <12:0>                          | U 15 |
|                 | : | 23791 |                                      |                                              |      |
|                 | : | 23792 |                                      |                                              |      |
| U 150D, 8872,EC | : | 23793 | JSR [DIAG.CODE]                      | ;FORCE THE IDC TO DIAG.IDLE                  | U 15 |
|                 | : | 23794 |                                      |                                              |      |
|                 | : | 23795 | LOOP.T2B.1:                          |                                              |      |
|                 | : | 23796 | LOOP.T2B.2:                          |                                              |      |
|                 | : | 23797 | LOOP.T2B.3:                          |                                              |      |
| U 150E, B640,15 | : | 23798 | MOV LS[#1] TO WR[0]                  | ;SET WR0 = 1                                 | U 15 |
| U 150F, BE82,15 | : | 23799 | MOV WR[0] TO LS[ERROR.NUMBER]        | ;SET UP ERROR NUMBER = 1 IN LS               | U 15 |
|                 | : | 23800 |                                      |                                              | U 15 |
| U 1510, B7AB,95 | : | 23801 | MOV LS[#6837] TO WR[3]               | ;SAVE THE EXPECTED DATA WR[3]                | U 15 |
| U 1511, B712,15 | : | 23802 | MOV LS[SETCSR.CRDY] TO WR[0]         | ;SET UP THE CSR TO BRANCH TO                 | U 15 |
| U 1512, B720,95 | : | 23803 | MOV LS[SETCSR.MAINT] TO WR[1]        | ; THE CORRECT IDC ROUTINE                    |      |
| U 1513, AEC2,15 | : | 23804 | BIS WR[1] TO WR[0]                   |                                              |      |
| U 1514, 370C,95 | : | 23805 | MOV LS[SETCSR.F6] TO WR[1]           |                                              | U 15 |
| U 1515, AEC2,15 | : | 23806 | BIS WR[1] TO WR[0]                   |                                              | U 15 |
| U 1516, 17A0,15 | : | 23807 | PORT.WRITE PORT.ADRS[CSR] WITH WR[0] | ;WRITE THE CSR                               |      |
|                 | : | 23808 |                                      | ;MAINT=1,CRDY=1,F<2:0>=110                   | U 15 |
|                 | : | 23809 | :UCLR ECC                            |                                              | U 15 |
|                 | : | 23810 | -----                                |                                              | U 15 |
|                 | : | 23811 | -----                                |                                              | U 15 |
|                 | : | 23812 | *****                                |                                              | U 15 |
|                 | : | 23813 | * IDC *                              |                                              | U 15 |
|                 | : | 23814 | *****                                |                                              | U 15 |
|                 | : | 23815 | -----                                |                                              |      |
|                 | : | 23816 | :018:                                |                                              |      |
|                 | : | 23817 | =000                                 |                                              |      |
|                 | : | 23818 | :DIAG.IDLE:                          |                                              |      |
|                 | : | 23819 |                                      |                                              |      |
|                 | : | 23820 | BEN0/F0,                             |                                              |      |
|                 | : | 23821 | BEN1/F1,                             | ;BRANCH ON THE CSR F<2:0> BITS               |      |
|                 | : | 23822 | BEN2/F2,                             |                                              |      |
|                 | : | 23823 | NAD/DIAG.IDLE                        |                                              |      |
|                 | : | 23824 | -----                                |                                              |      |
|                 | : | 23825 | :01E:                                |                                              |      |
|                 | : | 23826 | =110                                 |                                              |      |
|                 | : | 23827 |                                      |                                              |      |
|                 | : | 23828 | BEN0/CRDY.L,                         |                                              |      |
|                 | : | 23829 | BEN2/MAINT,                          | ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE |      |
|                 | : | 23830 | NAD/TEST.FUNC6                       | ; DEPENDENT ON CSR <CRDY> AND <MAINT>        |      |
|                 | : | 23831 | -----                                |                                              |      |
|                 | : | 23832 | :076:                                |                                              |      |
|                 | : | 23833 | =1*0                                 |                                              |      |
|                 | : | 23834 | :TEST.ECC.1:                         |                                              |      |
|                 | : | 23835 |                                      |                                              |      |
|                 | : | 23836 | FUNC/CLR.ECC,                        | ;ISSUE CLR ECC TO INIT THE DC007             |      |
|                 | : | 23837 | NAD/XFER.REQ                         | ;FLAG THE CPU                                |      |
|                 | : | 23838 | -----                                |                                              |      |
|                 | : | 23839 | :1D8:                                |                                              |      |
|                 | : | 23840 | :XFER.REQ:                           |                                              |      |
|                 | : | 23841 | UCMD/SET.XFER.RQST,                  | ;SEND XFER REQ TO THE CPU                    |      |
|                 | : | 23842 | NAD/DIAG.WAIT                        |                                              |      |
|                 | : | 23843 | -----                                |                                              |      |
|                 | : | 23844 | :1D7:                                |                                              |      |

```

23845 :DIAG.WAIT:
23846 :
23847 : BEN1/XFER.REQ ;WAIT FOR THE CPU TO ISSUE XFER GRANT
23848 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
23849 :-----
23850 :159:
23851 :=*0*
23852 :DIAG.WAIT1:
23853 :
23854 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
23855 : ; COMMAND FROM THE CPU
23856 :-----
23857 :15B:
23858 :=*1*
23859 :DIAG.WAIT2:
23860 :
23861 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
23862 : ; WAIT SOME MORE
23863 :-----
23864 :WAIT.IDC.T2B.1:
23865 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ FROM IDC
23866 : JMP [IDC.DONE.T2B.1] ;XFER REQ SET
23867 : JMP [WAIT.IDC.T2B.1] ;WAIT SOME MORE
23868 :
23869 :
23870 :IDC.DONE.T2B.1:
23871 : MOV LS[SETCSR.F0] TO WR[0] ;CLEAR CSR F<2:0>
23872 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
23873 : MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
23874 :
23875 :READ.POS.T2B.1:
23876 : MISC [SET.PORT.1.0] ;SELECT THE IDC
23877 : PORT.READ PORT.ADRS[POSITION] ;SET UP TO READ THE POSITION
23878 : MOV PORT TO LS[PORT0] ;READ THE POSITION AND
23879 : ; SAVE IT IN LS
23880 : MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
23881 : MOV WR[3] TO WR[1] ;SET UP THE EXPECTED DATA
23882 : JSR [CHECK.RESULT] ;CHECK THE DATA
23883 : JMP [LOOP.T2B.1] ;LOOP HERE IF ERR & LOE IS SET
23884 :
23885 : MOV LS[#2] TO WR[0]
23886 : MOV WR[0] TO LS[ERROR.NUMBER] ;ERROR 2
23887 :
23888 : INC WR[3] ;INCR THE EXPECTED DATA
23889 :
23890 : PORT.CONTROL [SEL.FIFO.A] ;SELECT FIFO A
23891 : PORT.CONTROL [CLEAR.FIFO.CNTR] ;SET ADRS = 0
23892 : MOV LS[#1] TO WR[0] ;GET THE DATA TO WRITE
23893 : PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;WRITE A ONE TO FIFO A
23894 : PORT.CONTROL [CLEAR.FIFO.CNTR] ;SET ADRS = 0
23895 :
23896 : MOV LS[SETCSR.CRDY] TO WR[1] ;SET UP THE CSR TO BRANCH TO
23897 : MOV LS[SETCSR.F7] TO WR[0] ; THE CORRECT IDC ROUTINE
23898 : BIS WR[1] TO WR[0]
23899 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
23900 : MAINT=0,CRDY=1,F<2:0>=111

```

U 1517, DB00,1A  
 U 1518, 8951,A4  
 U 1519, 0951,74

U 151A, B700,15  
 U 151B, 17A0,15  
 U 151C, 90FA,15

U 151D, 9090,15  
 U 151E, 9794,15  
 U 151F, 3832,15

U 1520, 3732,15  
 U 1521, 2006,95  
 U 1522, 0869,3C  
 U 1523, 8950,E4

U 1524, 3642,15  
 U 1525, BE82,15

U 1526, 2047,95

U 1527, 1708,15  
 U 1528, 17C0,15  
 U 1529, B640,15  
 U 152A, 97A8,15  
 U 152B, 17C0,15

U 152C, 3712,95  
 U 152D, 370E,15  
 U 152E, AEC2,15  
 U 152F, 17A0,15

ZZ-E  
 :  
 :

U 15  
 U 15  
 U 15

U 15  
 U 15  
 U 15

U 15  
 U 15  
 U 15  
 U 15

U 15  
 U 15  
 U 15  
 U 15  
 U 15  
 U 15

ZZ-ENKCF-1.4  
: ENKCF.MCR  
: ENKCF.MIC

ENKCF.MIC

MICRO2 1M(01)

Test 2B DC007 POS115-MAR-83

15-MAR-83 11:46:22

ENKCF Micro-diagnostic for IDC module

Sequence 491

Page 485

Test 2B DC007 POSITION REGISTER TEST(M8388 IDC MODULE)

```
:23900
:23901 ;CONTINUE BRANCHING
:23902
U 1530, B79F,15 :23903 MOV LS[SETCSR.R80] TO WR[2] ;CONTINUE TO BRANCH TO THE
U 1531, 3702,15 :23904 MOV LS[SETCSR.F1] TO WR[0] ; IDC ROUTINE
U 1532, AEC2,15 :23905 BIS WR[1] TO WR[0] ;SET CRDY TO HOLD SDIO LO
U 1533, AEC4,15 :23906 BIS WR[2] TO WR[0]
U 1534, 17A0,15 :23907 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:23908 ;R80=1,F<2:0>=001
:23909 -----
:23910 -----
:23911 *****
:23912 * IDC *
:23913 *****
:23914 -----
:23915 :018:
:23916 :=000
:23917 :DIAG.IDLE:
:23918
:23919 BEN0/F0,
:23920 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:23921 BEN2/F2,
:23922 NAD/DIAG.IDLE
:23923 -----
:23924 :01F:
:23925 :=111
:23926
:23927 BEN0/CRDY.L,
:23928 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:23929 NAD/TEST.FUNC7 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:23930 -----
:23931 :078:
:23932 :=0*0
:23933 :TEST.FUNC7:
:23934
:23935 NAD/GROUP0
:23936 -----
:23937 :163:
:23938 :=0**
:23939 :GROUP0:
:23940
:23941 BEN2/R80.FORMAT.H, ;WHEN THE R80 FORMAT BIT SETS..BRANCH
:23942 NAD/GROUP0 ; TO NEXT WORD AND THEN BRANCH ON THE
:23943 ; FUNCTION BITS
:23944 -----
:23945 :167:
:23946 :=1**
:23947
:23948 BEN0/F0,
:23949 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:23950 BEN2/F2,
:23951 NAD/GROUP0.F0
:23952 -----
:23953 :021:
:23954 :=001
```

ZZ-ENKCF-1.4 :  
: ENKCF.MCR  
: ENKCF.MIC

ENKCF.MIC

MICRO2 1M(01)

Test 2B DC007 POS115-MAR-83

15-MAR-83 11:46:22

Fiche 3 Frame C7

Sequence 492

Rev 01.40

Page 486

Test 2B DC007 POSITION REGISTER TEST(M8388 IDC MODULE)

```
:23955 :POS.REG.1:
:23956 :
:23957 : UCMD/SET.FIFOA, ;SEL FIFO A
:23958 : NAD/POS.REG.2
:23959 :-----
:23960 :1E9:
:23961 :POS.REG.2:
:23962 :
:23963 : ENB.FIFO/ON, ;LOAD THE DSR WITH DATA FROM THE FIFO
:23964 : LOAD.DSR/ON,
:23965 : NAD/POS.REG.3
:23966 :-----
:23967 :1EA:
:23968 :POS.REG.3:
:23969 :
:23970 : UMAINT/ON, ;ENABLE DSRO AS NRZ DATA
:23971 : CHECK.FIELD/ON, ;MAKE DC007 BELIEVE IT HAS READ THE
:23972 : MODE/CRC, ; CHECK FIELD
:23973 : NAD/POS.REG.4
:23974 :-----
:23975 :1EB:
:23976 :POS.REG.4:
:23977 :
:23978 : MODE/CRC, ;DROP CHECK FIELD WHILE THE INTERNAL
:23979 : ; DC007 SHIFT REG IS NON-ZERO TO
:23980 : ; ASSERT ERROR INTERNAL TO THE DC007
:23981 : ;ERROR ALLOWS THE POS REG TO INCR
:23982 : DATA.FIELD/ON, ;ASSERT DATA FIELD TO ALLOW CLK
:23983 : UMAINT/ON, ;TO SELECT DSRO AS NRZ DATA
:23984 : NAD/POS.REG.5
:23985 :-----
:23986 :1EC:
:23987 :POS.REG.5:
:23988 :
:23989 : DATA.FIELD/ON, ;TO ALLOW THE POS REG TO CLK ONCE
:23990 : MODE/CRC, ;TO PREVENT INTERNAL DC007 SIGNALS
:23991 : UMAINT/ON, ;TO SEL DSRO AS NRZ DATA
:23992 : NAD/XFER.REQ ;FLAG THE CPU
:23993 :-----
:23994 :1D8:
:23995 :XFER.REQ:
:23996 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:23997 : NAD/DIAG.WAIT
:23998 :-----
:23999 :1D7:
:24000 :DIAG.WAIT:
:24001 :
:24002 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:24003 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:24004 :-----
:24005 :159:
:24006 :=*0*
:24007 :DIAG.WAIT1:
:24008 :
:24009 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
```

```

:24010 : ; COMMAND FROM THE CPU
:24011 : -----
:24012 : 15B:
:24013 : =*1*
:24014 : DIAG.WAIT2:
:24015 :
:24016 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:24017 : ; WAIT SOME MORE
:24018 : -----
:24019 : WAIT.IDC.T2B.2:
U 1535, DB00,1A :24020 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ FROM IDC
U 1536, 8953,84 :24021 : JMP [IDC.DONE.T2B.2] ;XFER REQ SET
U 1537, 0953,54 :24022 : JMP [WAIT.IDC.T2B.2] ;WAIT SOME MORE
:24023 :
:24024 : IDC.DONE.T2B.2:
U 1538, B700,15 :24025 : MOV LS[SETCSR.F0] TO WR[0] ;CLEAR CSR F<2:0>
U 1539, 17A0,15 :24026 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
U 153A, 90FA,15 :24027 : MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
:24028 :
:24029 :
:24030 : READ.POS.T2B.2:
U 153B, 9090,15 :24031 : MISC [SET.PORT.1.0] ;SELECT THE IDC
U 153C, 9794,15 :24032 : PORT.READ PORT.ADRS[POSITION] ;SET UP TO READ THE POSITION
U 153D, 3B32,15 :24033 : MOV PORT TO LS[PORT0] ;READ THE POSITION AND
:24034 : ; SAVE IT IN LS
U 153E, 3732,15 :24035 : MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
U 153F, 2006,95 :24036 : MOV WR[3] TO WR[1] ;SET UP THE EXPECTED DATA
U 1540, 0869,3C :24037 : JSR [CHECK.RESULT] ;CHECK THE DATA
U 1541, 8950,E4 :24038 : JMP [LOOP.T2B.2] ;LOOP HERE IF ERR & LOE IS SET
:24039 :
:24040 : TEST.POSREG.T2B:
:24041 :
:24042 : ;A ONE IN WR[2] INDICATES THAT THE LIMIT HAS NOT BEEN REACHED
:24043 :
U 1542, 2F85,15 :24044 : CLR WR[2] ;CLR THE LIMIT INDICATOR
:24045 :
U 1543, 37B0,15 :24046 : MOV LS[#1021] TO WR[0] ;GET THE LIMIT VALUE
:24047 : CMP WR[3] WITH WR[0], ;LIMIT BEEN REACHED?
U 1544, A646,35 :24048 : DT(LONG)&SET.ALU.CC
U 1545, 8954,89 :24049 : JMP.IF[EQL] TO [CHECK.NXT.T2B] ;DO NOT INCR EXP DATA IF LIMIT
:24050 :
U 1546, 2047,95 :24051 : INC WR[3] ;INC THE EXPECTED DATA
U 1547, 3641,15 :24052 : MOV LS[#1] TO WR[2] ;LIMIT INDICATOR...1=NOT LIMIT
:24053 : CHECK.NXT.T2B:
:24054 : TST WR[2], ;LIMIT?
U 1548, 2005,35 :24055 : DT(LONG)&SET.ALU.CC
U 1549, 8954,C1 :24056 : JMP.IF[NEQ] TO [CONT.T2B] ;NOT LIMIT..STILL ERROR 2
:24057 :
U 154A, B788,15 :24058 : MOV LS[#3] TO WR[0]
U 154B, BE82,15 :24059 : MOV WR[0] TO LS[ERROR.NUMBER] ;LIMIT...ERROR 3
:24060 :
:24061 : CONT.T2B:
U 154C, 3712,95 :24062 : MOV LS[SETCSR.CRDY] TO WR[1] ;SET UP THE CSR TO BRANCH TO
U 154D, 370E,15 :24063 : MOV LS[SETCSR.F7] TO WR[0] ; THE CORRECT IDC ROUTINE
U 154E, AEC2,15 :24064 : BIS WR[1] TO WR[0]

```

```

U 154F, 17A0,15 :24065 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
 :24066 ;MAINT=0,CRDY=1,F<2:0>=111
 :24067
 :24068 ;CONTINUE BRANCHING
 :24069
U 1550, 379E,15 :24070 MOV LS[SETCSR.R80] TO WR[0] ;CONTINUE TO BRANCH
U 1551, AEC2,15 :24071 BIS WR[1] TO WR[0] ;SET CRDY TO HOLD SDOI LO
U 1552, B704,95 :24072 MOV LS[SETCSR.F2] TO WR[1] ;
U 1553, AEC2,15 :24073 BIS WR[1] TO WR[0]
U 1554, 17A0,15 :24074 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
 :24075 ;R80=1,F<2:0>=010
 :24076 -----
 :24077 -----
 :24078 *****
 :24079 * IDC *
 :24080 *****
 :24081 -----
 :24082 ;018:
 :24083 ;=000
 :24084 ;DIAG.IDLE:
 :24085
 :24086 BEN0/F0,
 :24087 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
 :24088 BEN2/F2,
 :24089 NAD/DIAG.IDLE
 :24090 -----
 :24091 ;01F:
 :24092 ;=111
 :24093
 :24094 BEN0/CRDY.L,
 :24095 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
 :24096 NAD/TEST.FUNC7 ;DEPENDENT ON CSR <CRDY> AND <MAINT>
 :24097 -----
 :24098 ;078:
 :24099 ;=0*0
 :24100 ;TEST.FUNC7:
 :24101
 :24102 NAD/GROUP0
 :24103 -----
 :24104 ;163:
 :24105 ;=0**
 :24106 ;GROUP0:
 :24107
 :24108 BEN2/R80.FORMAT.H, ;WHEN THE R80 FORMAT BIT SETS..BRANCH
 :24109 NAD/GROUP0 ;TO NEXT WORD AND THEN BRANCH ON THE
 :24110 ;FUNCTION BITS
 :24111 -----
 :24112 ;167:
 :24113 ;=1**
 :24114
 :24115 BEN0/F0,
 :24116 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
 :24117 BEN2/F2,
 :24118 NAD/GROUP0.F0
 :24119 -----

```

22-  
:  
:

[illegible]



```

U 1562, 2005,35 :24175 DT(LONG)&SET.ALU.CC
U 1563, 8957,19 :24176 JMP.IF[EQL] TO [TEST.STAT.T2B] ;JMP IF LIMIT
 :24177
 :24178 CMP LS[FFFF] WITH WR[3], ;TIME TO RESET EXPECTED DATA?
U 1564, CE29,B5 :24179 DT(LONG)&SET.ALU.CC
U 1565, 0954,21 :24180 JMP.IF[NEQ] TO [TEST.POSREG.T2B] ;JMP IF NOT TIME TO RESET
U 1566, 2F87,95 :24181 CLR WR[3] ;RESET THE EXPECTED DATA
U 1567, 3712,95 :24182 MOV LS[SETCSR.CRDY] TO WR[1] ;SET UP THE CSR TO BRANCH TO
U 1568, 370E,15 :24183 MOV LS[SETCSR.F7] TO WR[0] ; THE CORRECT IDC ROUTINE
U 1569, AEC2,15 :24184 BIS WR[1] TO WR[0]
U 156A, 17A0,15 :24185 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
 :24186 ;MAINT=0,CRDY=1,F<2:0>=111
 :24187
 :24188 ;CONTINUE BRANCHING
 :24189
U 156B, 379E,15 :24190 MOV LS[SETCSR.R80] TO WR[0] ;CONTINUE TO BRANCH
U 156C, AEC2,15 :24191 BIS WR[1] TO WR[0] ;SET CRDY TO HOLD SDIO LO
U 156D, 370A,95 :24192 MOV LS[SETCSR.F5] TO WR[1]
U 156E, AEC2,15 :24193 BIS WR[1] TO WR[0]
U 156F, 17A0,15 :24194 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
 :24195 ;R80=1,F<2:0>=101
 :24196 -----
 :24197 -----
 :24198 *****
 :24199 * IDC *
 :24200 *****
 :24201 -----
 :24202 :018:
 :24203 :=000
 :24204 :DIAG.IDLE:
 :24205
 :24206 BEN0/F0,
 :24207 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
 :24208 BEN2/F2,
 :24209 NAD/DIAG.IDLE
 :24210 -----
 :24211 :01F:
 :24212 :=111
 :24213
 :24214 BEN0/CRDY.L,
 :24215 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
 :24216 NAD/TEST.FUNC7 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
 :24217 -----
 :24218 :078:
 :24219 :=0*0
 :24220 :TEST.FUNC7:
 :24221
 :24222 NAD/GROUP0
 :24223 -----
 :24224 :163:
 :24225 :=0**
 :24226 :GROUP0:
 :24227
 :24228 BEN2/R80.FORMAT.H, ;WHEN THE R80 FORMAT BIT SETS..BRANCH
 :24229 NAD/GROUP0 ; TO NEXT WORD AND THEN BRANCH ON THE

```

```

:24230 : ; FUNCTION BITS
:24231 : -----
:24232 : 167:
:24233 : =1**
:24234 :
:24235 : BEN0/F0,
:24236 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:24237 : BEN2/F2,
:24238 : NAD/GROUP0.F0
:24239 : -----
:24240 : 025:
:24241 : =101
:24242 : SHFT.CLK.POS.1:
:24243 :
:24244 : UCMD/SET.FIFOA, ;SEL FIFO A
:24245 : NAD/SHFT.CLK.POS.2
:24246 : -----
:24247 : 1ED:
:24248 : SHFT.CLK.POS.2:
:24249 :
:24250 : ENB.FIFO/ON, ;LOAD THE DSR WITH DATA FROM THE FIFO
:24251 : LOAD.DSR/ON,
:24252 : NAD/SHFT.CLK.POS.3
:24253 : -----
:24254 : 1EE:
:24255 : SHFT.CLK.POS.3:
:24256 :
:24257 : UMAINT/ON, ;ENABLE DSRO AS NRZ DATA
:24258 : DATA.FIELD/ON, ;TO ALLOW THE POS REG TO CLK ONCE
:24259 : MODE/CRC, ;TO PREVENT INTERNAL DC007 SIGNALS
:24260 : NAD/XFER.REQ ;FLAG THE CPU
:24261 : -----
:24262 : 1D8:
:24263 : XFER.REQ:
:24264 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:24265 : NAD/DIAG.WAIT
:24266 : -----
:24267 : 1D7:
:24268 : DIAG.WAIT:
:24269 :
:24270 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:24271 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:24272 : -----
:24273 : 159:
:24274 : =*0*
:24275 : DIAG.WAIT1:
:24276 :
:24277 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:24278 : ; COMMAND FROM THE CPU
:24279 : -----
:24280 : 15B:
:24281 : =*1*
:24282 : DIAG.WAIT2:
:24283 :
:24284 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....

```

[illegible]

:24314 .PAGE "Test 2C DC007 WRITE CLK AND CRC MODE TEST (M8388 IDC MODULE)"  
:24315 :++  
:24316 :  
:24317 : Test Description:  
:24318 :  
:24319 : This test is designed to test that the write clk of the DC007 operates  
:24320 : correctly and also that the internal DC007 pattern register shifts  
:24321 : correctly when in CRC mode. This will be accomplished by first  
:24322 : writing a one into the pattern register. The write clk and CRC mode  
:24323 : will then be selected and the register will be shifted once. The CPU  
:24324 : will then check the contents of the pattern register. This will be done  
:24325 : 31 times to assure that the register is operational.  
:24326 : Only bits <12:00> of the pattern register can be read by the CPU.  
:24327 :  
:24328 : Assumptions:  
:24329 :  
:24330 : It is assumed that all previous tests have been run successfully.  
:24331 :  
:24332 : Test Steps:  
:24333 :  
:24334 : 1. Set up error mask, error number, and module number in LS  
:24335 : (for error printouts) and clear previous error number in LS.  
:24336 : 2. Force the IDC microcode to DIAG.IDLE.  
:24337 : 3. Select an RL02 disk or a line with no drive attached to assert  
:24338 : RL02 H which will select the 4.1 MHZ clk as SYS CLK.  
:24339 : 4. Write a one to FIFO A address 0.  
:24340 : 5. Set up the CSR to branch to the correct IDC routine.  
:24341 :  
:24342 :  
:24343 : \*\*\*\*\*  
:24344 : \* IDC \*  
:24345 : \*\*\*\*\*  
:24346 :  
:24347 : IDC1. Issue a UCLR ECC to zero the pattern register.  
:24348 : IDC2. Shift a one into the pattern register.  
:24349 : IDC3. Send XFER REQ to flag the CPU.  
:24350 :  
:24351 : 6. Wait for a XFER REQ from IDC.  
:24352 : 7. Clear CSR F<2:0> and send XFER GRANT.  
:24353 : 8. Check the data in the pattern register.  
:24354 : 9. Set up the CSR to branch to the correct IDC routine.  
:24355 :  
:24356 :  
:24357 : \*\*\*\*\*  
:24358 : \* IDC \*  
:24359 : \*\*\*\*\*  
:24360 :  
:24361 : IDC4. Select SYS CLK.  
:24362 : IDC5. Assert DATA FIELD and UDISK WRITE to select the WRT  
:24363 : CLK and allow the pattern reg to shift once.  
:24364 : IDC6. Select CPU CLK.  
:24365 : IDC7. Send XFER REQ to flag the CPU.  
:24366 :  
:24367 : 10. Wait for a XFER REQ from IDC.  
:24368 : 11. Clear CSR F<2:0> and send XFER GRANT.

:24369 : 12. Read PATTERN and check the data  
 :24370 : 13. Repeat Steps 9 - 13 31 times.  
 :24371 : 14. Clear IDC.  
 :24372 : 15. End test.

:24373 : Errors:

|  | Printout:          |                    |
|--|--------------------|--------------------|
|  | EXPECTED           | RECEIVED           |
|  | INTERNAL SHIFT REG | INTERNAL SHIFT REG |
|  | (ECC PATTERN)      | (ECC PATTERN)      |

:24379 : ERROR 1 - The CRC/ECC chip internal Pattern register failed in CRC mode.

:24383 : ERROR 2 - Either the SYS CLK select logic failed, or the SYS CLK failed, or the CRC/ECC internal Pattern register failed when using the WRT CLK in CRC mode.

:24387 : Debug:

:24389 : ERROR 1 -

:24391 : The IDC microcode will execute an instruction with FUNC <3:0>=0001.  
 :24392 : These bits will be decoded and will assert UCLR ECC L. This signal  
 :24393 : is the Clear line to the CRC/ECC chip and will clear the CRC/ECC  
 :24394 : chip internal Pattern register.  
 :24395 : The data pattern = 1 is loaded into the DSR. The next IDC micro-  
 :24396 : instruction will assert UMAINT H, UDATA FIELD H and UENB CRC H.  
 :24397 : UMAINT H will allow DSR 0 to become NRZ WRT DATA H which is enabled  
 :24398 : to shift into the Pattern register by UDATA FIELD H = H. UENB CRC H = H  
 :24399 : allows only bits <15:0> of the Pattern register to be used and a  
 :24400 : different XORing scheme is used than when in ECC mode. The CPU will  
 :24401 : then read the Pattern register bits <21:31>. Although these bits are  
 :24402 : not used in CRC mode, this allows certain internal conditions to be  
 :24403 : tested.

:24404 : ERROR 2 -

:24405 : The IDC will execute an instruction to select the System clock.  
 :24406 : USEL SYS H = H  
 :24407 : UCHANGE CLKSRC H = H  
 :24408 : UCHANGE CLKSRC H = H is gated with END SYNC CLR L = H and clears the  
 :24409 : CLOCK SYNC FF.  
 :24410 : When CURRENT CLK L, UCHANGE CLKSRC H is latch through a FF and asserts  
 :24411 : END SYNC CLR H and END SYNC CLR L.  
 :24412 : END SYNC CLR H = H is gated with CURRENT CLK H and will clock the  
 :24413 : Clock sequencer FF, thus latching up USEL SYS CLK H as  
 :24414 : SEL SYS CLK H = H. When SEL SYS CLK H = H, SYS CLK H becomes CURRENT  
 :24415 : CLK H and is inverted to become CURRENT CLK L.  
 :24416 : Simultaneously, CURRENT CLK H is clocking the CLOCK SYNC FF. On the  
 :24417 : third CURRENT CLK H, SYNC FF 2 H is asserted. This is inverted to  
 :24418 : SYNC FF 2 L. SYNC FF 2 L is latch by CURRENT CLK L as ENB SEQ CLK L=L  
 :24419 : and ENB SEQ CLK H = H. SEL SYS CLK H = H is gated with ENB SEQ CLK H=H  
 :24420 : and allows SYS CLK H to become SEQ CLK H.  
 :24421 : RLO2 H = H is the SEL line to a MUX which will allow 4.1 MHZ CLK L  
 :24422 : to become SYS CLK H. 8.2 MHZ CLK H is divided by the PEAK SHFT COMP  
 :24423 :

U 16  
U 16  
U 16  
  
U 16  
U 16  
U 16  
  
U 16  
U 16  
  
U 16  
U 16  
U 16  
U 16  
U 16  
U 16

```

ZZ-ENKCF-1.4 : ENKCF.MIC Test 2C DC007 WRIT15-MAR-83 M 7 Fiche 3 Frame M7 Sequence 502 Page 496 ZZ-E
: ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40
: ENKCF.MIC Test 2C DC007 WRITE CLK AND CRC MODE TEST (M8388 IDC MODULE)

U 158F, 8959,14 :24479 JMP [TEST.T2C.1]
:24480 SET.DS0.T2C.1:
U 1590, 3715,15 :24481 MOV LS[SETCSR.DS0] TO WR[2] ;SAVE THE DATA TO SEL DRIVE 0
:24482 ; IN WR[2]
:24483 TEST.T2C.1:
:24484 LOOP.T2C.2:
:24485
U 1591, B7B3,95 :24486 MOV LS[#A0010500] TO WR[3] ;EXPECTED DATA
U 1592, 17D8,15 :24487 PORT.CONTROL [SEL.FIFO.A] ;SELECT FIFO A
U 1593, 17C0,15 :24488 PORT.CONTROL [CLEAR.FIFO.CNTR] ;SET FIFO A ADRS = 0
U 1594, B640,15 :24489 MOV LS[#1] TO WR[0] ;GET THE DATA TO WRITE
U 1595, 97A8,15 :24490 PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;WRITE A ONE TO FIFO A
U 1596, 17C0,15 :24491 PORT.CONTROL [CLEAR.FIFO.CNTR] ;SET FIFO A ADRS = 0
:24492
:24493 LOOP.T2C.1:
U 1597, B640,15 :24494 MOV LS[#1] TO WR[0] ;SET WRO = 1
U 1598, BE82,15 :24495 MOV WR[0] TO LS[ERROR.NUMBER] ;SET UP ERROR NUMBER = 1 IN LS
:24496
U 1599, B712,15 :24497 MOV LS[SETCSR.CRDY] TO WR[0] ;SET UP THE CSR TO BRANCH TO
U 159A, B720,95 :24498 MOV LS[SETCSR.MAINT] TO WR[1] ; THE CORRECT IDC ROUTINE
U 159B, AEC2,15 :24499 BIS WR[1] TO WR[0]
U 159C, 370C,95 :24500 MOV LS[SETCSR.F6] TO WR[1]
U 159D, AEC2,15 :24501 BIS WR[1] TO WR[0]
U 159E, 17A0,15 :24502 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:24503 ;MAINT=1,CRDY=1,F<2:0>=110
:24504 ;UCLR ECC
:24505 -----
:24506 -----
:24507 : *****
:24508 : * IDC *
:24509 : , *****
:24510 -----
:24511 :018:
:24512 :=000
:24513 :DIAG.IDLE:
:24514 :
:24515 : BEN0/F0,
:24516 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:24517 : BEN2/F2,
:24518 : NAD/DIAG.IDLE
:24519 -----
:24520 :01E:
:24521 :=110
:24522 :
:24523 : BEN0/CRDY.L,
:24524 : BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:24525 : NAD/TEST.FUNC6 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:24526 -----
:24527 :076:
:24528 :=1*0
:24529 :TEST.ECC.1:
:24530 :
:24531 : FUNC/CLR.ECC, ;ISSUE CLR ECC TO INIT THE DC007
:24532 : NAD/XFER.REQ ;FLAG THE CPU
:24533 -----

```

|              |   |           |                                                              |                                       |              |          |      |
|--------------|---|-----------|--------------------------------------------------------------|---------------------------------------|--------------|----------|------|
| ZZ-ENKCF-1.4 | : | ENKCF.MIC | Test 2C DC007 WRIT15-MAR-83                                  | Fiche 3 Frame N7                      | Sequence 503 | Page 497 | ZZ-E |
| :            | : | ENKCF.MCR | MICRO2 1M(01) 15-MAR-83 11:46:22                             | ENKCF Micro-diagnostic for IDC module | Rev 01.40    |          | :    |
| :            | : | ENKCF.MIC | Test 2C DC007 WRITE CLK AND CRC MODE TEST (M8388 IDC MODULE) |                                       |              |          | :    |

  

|                 |   |       |   |                                      |  |                                               |      |
|-----------------|---|-------|---|--------------------------------------|--|-----------------------------------------------|------|
|                 | : | 24534 | : | 1D8:                                 |  |                                               |      |
|                 | : | 24535 | : | XFER.REQ:                            |  |                                               |      |
|                 | : | 24536 | : | UCMD/SET.XFER.RQST,                  |  | ;SEND XFER REQ TO THE CPU                     |      |
|                 | : | 24537 | : | NAD/DIAG.WAIT                        |  |                                               |      |
|                 | : | 24538 | : | -----                                |  |                                               |      |
|                 | : | 24539 | : | 1D7:                                 |  |                                               |      |
|                 | : | 24540 | : | DIAG.WAIT:                           |  |                                               |      |
|                 | : | 24541 | : |                                      |  |                                               |      |
|                 | : | 24542 | : | BEN1/XFER.REQ,                       |  | ;WAIT FOR THE CPU TO ISSUE XFER GRANT         |      |
|                 | : | 24543 | : | NAD/DIAG.WAIT1                       |  | ; BEFORE RETURNING TO DIAG.IDLE               |      |
|                 | : | 24544 | : | -----                                |  |                                               |      |
|                 | : | 24545 | : | 159:                                 |  |                                               |      |
|                 | : | 24546 | : | =*0*                                 |  |                                               |      |
|                 | : | 24547 | : | DIAG.WAIT1:                          |  |                                               |      |
|                 | : | 24548 | : |                                      |  |                                               |      |
|                 | : | 24549 | : | NAD/DIAG.IDLE                        |  | ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND |      |
|                 | : | 24550 | : |                                      |  | ; COMMAND FROM THE CPU                        |      |
|                 | : | 24551 | : | -----                                |  |                                               |      |
|                 | : | 24552 | : | 15B:                                 |  |                                               |      |
|                 | : | 24553 | : | =*1*                                 |  |                                               |      |
|                 | : | 24554 | : | DIAG.WAIT2:                          |  |                                               |      |
|                 | : | 24555 | : |                                      |  |                                               |      |
|                 | : | 24556 | : | NAD/DIAG.WAIT                        |  | ;XFER GRANT HAS NOT BEEN ISSUED....           |      |
|                 | : | 24557 | : |                                      |  | ; WAIT SOME MORE                              |      |
|                 | : | 24558 | : | -----                                |  |                                               |      |
|                 | : | 24559 | : | WAIT.IDC.T2C.1:                      |  |                                               |      |
| U 159F, DB00,1A | : | 24560 | : | SKIP.IF[NO.FAST.INTERRUPT]           |  | ;WAIT FOR XFER REQ FROM IDC                   |      |
| U 15A0, 895A,24 | : | 24561 | : | JMP [IDC.DONE.T2C.1]                 |  | ;XFER REQ SET                                 |      |
| U 15A1, 0959,F4 | : | 24562 | : | JMP [WAIT.IDC.T2C.1]                 |  | ;WAIT SOME MORE                               |      |
|                 | : | 24563 | : | -----                                |  |                                               |      |
|                 | : | 24564 | : | IDC.DONE.T2C.1:                      |  |                                               |      |
| U 15A2, B700,15 | : | 24565 | : | MOV LS[SETCSR.F0] TO WR[0]           |  | ;CLEAR CSR F<2:0>                             |      |
| U 15A3, 17A0,15 | : | 24566 | : | PORT.WRITE PORT.ADRS[CSR] WITH WR[0] |  | ;WRITE THE CSR                                |      |
| U 15A4, 90FA,15 | : | 24567 | : | MISC2 [TRANSFER.GRANT]               |  | ;SEND GRANT TO CLR THE XFER REQ               |      |
|                 | : | 24568 | : |                                      |  |                                               |      |
| U 15A5, 3712,95 | : | 24569 | : | MOV LS[SETCSR.CRDY] TO WR[1]         |  | ;SET UP THE CSR TO BRANCH TO                  |      |
| U 15A6, 370E,15 | : | 24570 | : | MOV LS[SETCSR.F7] TO WR[0]           |  | ; THE CORRECT IDC ROUTINE                     |      |
| U 15A7, AEC2,15 | : | 24571 | : | BIS WR[1] TO WR[0]                   |  |                                               | U 16 |
| U 15A8, 17A0,15 | : | 24572 | : | PORT.WRITE PORT.ADRS[CSR] WITH WR[0] |  | ;WRITE THE CSR                                | U 16 |
|                 | : | 24573 | : |                                      |  | ;MAINT=0,CRDY=1,F<2:0>=111                    |      |
|                 | : | 24574 | : | ;CONTINUE BRANCHING                  |  |                                               |      |
|                 | : | 24575 | : |                                      |  |                                               |      |
| U 15A9, 379E,15 | : | 24576 | : | MOV LS[SETCSR.R80] TO WR[0]          |  | ;CONTINUE TO BRANCH TO ROUTINE                | U 16 |
| U 15AA, 370A,95 | : | 24577 | : | MOV LS[SETCSR.F5] TO WR[1]           |  |                                               | U 16 |
| U 15AB, AEC2,15 | : | 24578 | : | BIS WR[1] TO WR[0]                   |  |                                               | U 16 |
| U 15AC, 3712,95 | : | 24579 | : | MOV LS[SETCSR.CRDY] TO WR[1]         |  | ;SET CRDY TO HOLD SDIO LOW                    |      |
| U 15AD, AEC2,15 | : | 24580 | : | BIS WR[1] TO WR[0]                   |  |                                               | U 16 |
| U 15AE, AEC4,15 | : | 24581 | : | BIS WR[2] TO WR[0]                   |  | ;SELECT AN RL02                               |      |
| U 15AF, 17A0,15 | : | 24582 | : | PORT.WRITE PORT.ADRS[CSR] WITH WR[0] |  | ;WRITE THE CSR                                |      |
|                 | : | 24583 | : |                                      |  | ;R80=1, F<2:0>=101                            |      |
|                 | : | 24584 | : | -----                                |  |                                               |      |
|                 | : | 24585 | : | -----                                |  |                                               |      |
|                 | : | 24586 | : |                                      |  |                                               |      |
|                 | : | 24587 | : | *****                                |  |                                               |      |
|                 | : | 24588 | : | * IDC *                              |  |                                               |      |
|                 | : |       | : | *****                                |  |                                               |      |



ZZ-ENKCF-1.4  
: ENKCF.MCR  
: ENKCF.MIC

ENKCF.MIC

MICRO2 1M(01)

Test 2C DC007 WRIT15-MAR-83

15-MAR-83 11:46:22

B 8

Fiche 3 Frame B8

Sequence 504

Rev 01.40

Page 498

Test 2C DC007 WRITE CLK AND CRC MODE TEST (M8388 IDC MODULE)

```
:24589 :-----
:24590 :018:
:24591 :=000
:24592 :DIAG.IDLE:
:24593 :
:24594 : BEN0/F0,
:24595 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:24596 : BEN2/F2,
:24597 : NAD/DIAG.IDLE
:24598 :-----
:24599 :01F:
:24600 :=111
:24601 :
:24602 : BEN0/CRDY.L,
:24603 : BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:24604 : NAD/TEST.FUNC7 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:24605 :-----
:24606 :078:
:24607 :=0*0
:24608 :TEST.FUNC7:
:24609 :
:24610 : NAD/GROUP0
:24611 :-----
:24612 :163:
:24613 :=0**
:24614 :GROUP0:
:24615 :
:24616 : BEN2/R80.FORMAT.H, ;WHEN THE R80 FORMAT BIT SETS..BRANCH
:24617 : NAD/GROUP0 ; TO NEXT WORD AND THEN BRANCH ON THE
:24618 : ; FUNCTION BITS
:24619 :-----
:24620 :167:
:24621 :=1**
:24622 :
:24623 : BEN0/F0,
:24624 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:24625 : BEN2/F2,
:24626 : NAD/GROUP0.F0
:24627 :-----
:24628 :025:
:24629 :=101
:24630 :SHFT.CLK.POS.1:
:24631 :
:24632 : UCMD/SET.FIFOA, ;SEL FIFO A
:24633 : NAD/SHFT.CLK.POS.2
:24634 :-----
:24635 :1ED:
:24636 :SHFT.CLK.POS.2:
:24637 :
:24638 : ENB.FIFO/ON, ;LOAD THE DSR WITH DATA FROM THE FIFO
:24639 : LOAD.DSR/ON,
:24640 : NAD/SHFT.CLK.POS.3
:24641 :-----
:24642 :1EE:
:24643 :SHFT.CLK.POS.3:
```

```

24644 :
24645 : UMAINT/ON, ;ENABLE DSRO AS NRZ DATA
24646 : DATA.FIELD/ON, ;TO ALLOW THE POS REQ TO CLK ONCE
24647 : MODE/CRC, ;TO PREVENT INTERNAL DC007 SIGNALS
24648 : NAD/XFER.REQ ;FLAG THE CPU
24649 :-----
24650 :1D8:
24651 :XFER.REQ:
24652 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
24653 : NAD/DIAG.WAIT
24654 :-----
24655 :1D7:
24656 :DIAG.WAIT:
24657 :
24658 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
24659 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
24660 :-----
24661 :159:
24662 :=*0*
24663 :DIAG.WAIT1:
24664 :
24665 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
24666 : ; COMMAND FROM THE CPU
24667 :-----
24668 :15B:
24669 :=*1*
24670 :DIAG.WAIT2:
24671 :
24672 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
24673 : ; WAIT SOME MORE
24674 :-----
24675 :WAIT.IDC.T2C.2:
24676 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ FROM IDC
24677 : JMP [IDC.DONE.T2C.2] ;XFER REQ SET
24678 : JMP [WAIT.IDC.T2C.2] ;WAIT SOME MORE
24679 :-----
24680 :IDC.DONE.T2C.2:
24681 : MOV LS[SETCSR.F0] TO WR[0] ;CLEAR CSR F<2:0>
24682 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
24683 : MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
24684 :-----
24685 :READ.PAT.T2C.1:
24686 : MISC [SET.PORT.I.0] ;SELECT THE IDC
24687 : PORT.READ PORT.ADRS[PATTERN] ;SET UP TO READ PATTERN REG
24688 : MOV PORT TO LS[PORT0] ;READ THE DATA FROM PAT AND
24689 : ; SAVE IT IN LS
24690 : MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
24691 : MOV WR[3] TO WR[1] ;SET UP EXPECTED DATA
24692 : JSR [CHECK.RESULT] ;CHECK THE DATA
24693 : JMP [LOOP.T2C.1] ;LOOP HERE IF ERR & LOE IS SET
24694 :-----
24695 : INC LS[ERROR.NUMBER] ;ERROR 2
24696 : MOV LS[#1F] TO WR[0]
24697 : MOV WR[0] TO LS[T8] ;USE AS A LOOP COUNTER=31
24698 :

```

U 1580, DB00,1A  
 U 1581, 895B,34  
 U 1582, 895B,04

U 1583, B700,15  
 U 1584, 17A0,15  
 U 1585, 90FA,15

U 1586, 9090,15  
 U 1587, 1790,15  
 U 1588, 3B32,15

U 1589, 3732,15  
 U 158A, 2006,95  
 U 158B, 0869,3C  
 U 158C, 8959,74

U 158D, FF82,15  
 U 158E, 37B6,15  
 U 158F, 3E10,15

```

ZZ-ENKCF-1.4 : ENKCF.MIC Test 2C DC007 WRIT15-MAR-83 D 8
: ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module
: ENKCF.MIC Test 2C DC007 WRITE CLK AND CRC MODE TEST (M8388 IDC MODULE)
Sequence 506 Page 500
Fiche 3 Frame D8

:24699 CHK.PAT.REG.T2C:
:24700
:24701 ;SET UP THE NEW EXPECTED DATA
U 15C0, A351,95 :24702 ASHR WR[3] ;SHIFT DATA ONCE RIGHT
:24703 BIT LS[BIT15] WITH WR[3], ;IS BIT 15 SET?
U 15C1, 595F,85 :24704 DT(LONG)&SET.ALU.CC
U 15C2, 095C,79 :24705 JMP.IF[BITS.CLR] TO [NO.XOR.T2C.1] ;JMP IF BIT15 = 0
:24706
U 15C3, 37B4,95 :24707 MOV LS[#20010500] TO WR[1] ;GET THE MASK
U 15C4, AF83,95 :24708 XOR WR[1] TO WR[3] ;CREATE THE NEW EXPECTED DATA
U 15C5, 477F,95 :24709 BIS LS[BIT31] TO WR[3]
U 15C6, 895C,84 :24710 JMP [SET.EXP.DAT.T2C]
:24711 NO.XOR.T2C.1:
U 15C7, C57F,95 :24712 BIC LS[BIT31] TO WR[3] ;CREATE THE NEW EXPECTED DATA
:24713 SET.EXP.DAT.T2C:
U 15C8, 455F,95 :24714 BIC LS[BIT15] TO WR[3] ;WR[3] HAS NEW EXPECTED DATA
:24715
U 15C9, 3712,95 :24716 MOV LS[SETCSR.CRDY] TO WR[1] ;SET UP THE CSR TO BRANCH TO
U 15CA, 370E,15 :24717 MOV LS[SETCSR.F7] TO WR[0] ; THE CORRECT IDC ROUTINE
U 15CB, AEC2,15 :24718 BIS WR[1] TO WR[0]
U 15CC, 17A0,15 :24719 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:24720 ;MAINT=0,CRDY=1,F<2:0>=111
:24721 ;CONTINUE BRANCHING
U 15CD, 379E,15 :24722 MOV LS[SETCSR.R80] TO WR[0] ;CONTINUE TO BRANCH
U 15CE, 370C,95 :24723 MOV LS[SETCSR.F6] TO WR[1]
U 15CF, AEC2,15 :24724 BIS WR[1] TO WR[0]
U 15D0, 3712,95 :24725 MOV LS[SETCSR.CRDY] TO WR[1] ;SET CRDY TO HOLD SDIO =LO
U 15D1, AEC2,15 :24726 BIS WR[1] TO WR[0]
U 15D2, AEC4,15 :24727 BIS WR[2] TO WR[0] ;SELECT AN RL02
U 15D3, 17A0,15 :24728 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:24729 ;R80=1,F<2:0>=110
:24730 -----
:24731 -----
:24732 *****
:24733 * IDC *
:24734 *****
:24735 -----
:24736 018:
:24737 =000
:24738 DIAG.IDLE:
:24739
:24740 BEN0/F0,
:24741 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:24742 BEN2/F2,
:24743 NAD/DIAG.IDLE
:24744 -----
:24745 01F:
:24746 =111
:24747
:24748 BEN0/CRDY.L,
:24749 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:24750 NAD/TEST.FUNC7 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:24751 -----
:24752 078:
:24753 =0*0

```

ZZ-ENKCF-1.4  
: ENKCF.MCR  
: ENKCF.MIC

ENKCF.MIC

MICRO2 1M(01)

Test 2C DC007 WRIT15-MAR-83

15-MAR-83 11:46:22

E 8

Fiche 3 Frame E8

Sequence 507

Rev 01.40

Page 501

Test 2C DC007 WRITE CLK AND CRC MODE TEST (M8388 IDC MODULE)

ZZ-  
:  
:

```
:24754 :TEST.FJNC7:
:24755 :
:24756 : NAD/GROUP0
:24757 :-----
:24758 :163:
:24759 :=0**
:24760 :GROUP0:
:24761 :
:24762 : BEN2/R80.FORMAT.H, ;WHEN THE R80 FORMAT BIT SETS..BRANCH
:24763 : NAD/GROUP0 ; TO NEXT WORD AND THEN BRANCH ON THE
:24764 : ; FUNCTION BITS
:24765 :-----
:24766 :167:
:24767 :=1**
:24768 :
:24769 : BEN0/F0,
:24770 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:24771 : BEN2/F2,
:24772 : NAD/GROUP0.F0
:24773 :-----
:24774 :026:
:24775 :=110
:24776 :CRC.PAT.1:
:24777 :
:24778 : ENB.SYS.CLK, ;SELECT THE RLO2 SYSTEM CLK
:24779 : NAD/CRC.PAT.2
:24780 :-----
:24781 :1EF:
:24782 :CRC.PAT.2:
:24783 :
:24784 : MODE/CRC, ;SELECT CRC TO SHIFT BIT 15
:24785 : DATA.FIELD/ON, ;DATA.FIELD AND DISK.WRITE ALLOW
:24786 : DISK.WRITE/ON, ; THE INTERNAL WRT CLK
:24787 : UMAINT/ON, ;SELECT DSRO AS NRZ DATA
:24788 : NAD/CRC.PAT.3
:24789 :-----
:24790 :1F0:
:24791 :CRC.PAT.3:
:24792 :
:24793 : ENB.CPU.CLK, ;RETURN TO THE CPU CLK
:24794 : NAD/XFER.REQ ;FLAG THE CPU
:24795 :-----
:24796 :1D8:
:24797 :XFER.REQ:
:24798 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:24799 : NAD/DIAG.WAIT
:24800 :-----
:24801 :1D7:
:24802 :DIAG.WAIT:
:24803 :
:24804 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:24805 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:24806 :-----
:24807 :159:
:24808 :=*0*
```

```

:24809 ;DIAG.WAIT1:
:24810 ;
:24811 ; NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:24812 ; COMMAND FROM THE CPU
:24813 -----
:24814 ;15B:
:24815 ;=*1*
:24816 ;DIAG.WAIT2:
:24817 ;
:24818 ; NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:24819 ; WAIT SOME MORE
:24820 -----
:24821 WAIT.IDC.T2C.3:
U 15D4, DB00,1A :24822 SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ FROM IDC
U 15D5, 095D,74 :24823 JMP [IDC.DONE.T2C.3] ;XFER REQ SET
U 15D6, 095D,44 :24824 JMP [WAIT.IDC.T2C.3] ;WAIT SOME MORE
:24825
:24826 IDC.DONE.T2C.3:
U 15D7, B700,15 :24827 MOV LS[SETCSR.F0] TO WR[0] ;CLEAR CSR F<2:0>
U 15D8, 17A0,15 :24828 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
U 15D9, 90FA,15 :24829 MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
:24830
:24831
:24832 READ.PAT.T2C.2:
U 15DA, 9090,15 :24833 MISC [SET.PORT.1.0] ;SELECT THE IDC
U 15DB, 1790,15 :24834 PORT.READ PORT.ADRS[PATTERN] ;SET UP TO READ PATTERN REG
U 15DC, 3B32,15 :24835 MOV PORT TO LS[PORT0] ;READ THE DATA FROM PAT AND
:24836 ; SAVE IT IN LS
U 15DD, 3732,15 :24837 MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
U 15DE, 2006,95 :24838 MOV WR[3] TO WR[1] ;SET UP EXPECTED DATA
U 15DF, 0869,3C :24839 JSR [CHECK.RESULT] ;CHECK THE DATA
U 15E0, 8959,14 :24840 JMP [LOOP.T2C.2] ;LOOP HERE IF ERR & LOE IS SET
:24841 ;DONE?
U 15E1, FC10,35 :24842 DT(LONG)&SET.ALU.CC
U 15E2, 095C,01 :24843 JMP.IF[NEQ] TO [CHK.PAT.REG.T2C] ;JMP IF NOT DONE
:24844
U 15E3, 8875,3C :24845 JSR [CLEANUP] ;CLEANUP THE CONDITIONS SET BY
:24846 ; THIS TEST
:24847 END.T2C: ;END TEST
:24848
:24849

```

```

24850 :PAGE "Test 2D CORRECTABLE ECC ERROR TEST (M8388 IDC MODULE)"
24851 :++
24852 :
24853 : Test Description:
24854 :
24855 : This test is designed to insure that the DC007 chip recognizes a
24856 : correctable error and that the ECC STAT <1:0> = 11 when a correctable
24857 : error is found.
24858 : This will be done by first issuing a UCLR ECC which will initialize the
24859 : DC007 chip. Then an error will be generated. A pattern is loaded into
24860 : internal DC007 shift pattern register that will then generate a
24861 : correctable error. The RL02 System clock will be used.
24862 :
24863 : Assumptions:
24864 :
24865 : It is assumed that all previous tests have run successfully.
24866 :
24867 : Test Steps:
24868 :
24869 : 1. Set up error mask, error number and module number in LS
24870 : (for error printouts) and clear the previous error number in LS.
24871 : 2. Force the IDC microcode to DIAG.IDLE.
24872 : 3. Select a drive number that is not an R80.
24873 : 4. Write a one to FIFO A address 0.
24874 : 5. Set up the CSR to branch to the correct IDC microcode routine.
24875 :
24876 :
24877 : *****
24878 : * IDC *
24879 : *****
24880 :
24881 : IDC1. Execute an instruction with FUNC <35:32> = 0001,
24882 : UCLR ECC.
24883 :
24884 : IDC2. Shift the one from FIFO A into the internal DC007 pattern
24885 : register.
24886 :
24887 : 6. Wait for a XFER REQ from the IDC.
24888 : 7. Clear CSR F<2:0> and send XFER GRANT.
24889 : 8. Set up the CSR to branch to the correct IDC routine.
24890 :
24891 :
24892 :
24893 : *****
24894 : * IDC *
24895 : *****
24896 :
24897 : IDC3. Assert DC007 ERR by first asserting CHECK.FIELD and then
24898 : deasserting CHECK.FIELD.
24899 :
24900 :
24901 : 9. Wait for a XFER REQ from the IDC.
24902 : 10. Clear CSR F<2:0> and send XFER GRANT to clr XFER REQ.
24903 : 11. Set up the CSR to branch to the correct IDC routine.
24904 :

```

```

24905
24906
24907
24908
24909
24910
24911
24912
24913
24914
24915
24916
24917
24918
24919
24920
24921
24922
24923
24924
24925
24926
24927
24928
24929
24930
24931
24932
24933
24934
24935
24936
24937
24938
24939
24940
24941
24942
24943
24944
24945
24946
24947
24948
24949
24950
24951
24952
24953
24954
24955
24956
24957
24958
24959

* IDC *

IDC4. Select the RLO2 SYSTEM clock.
IDC5. Load the Loop counter= 00.
IDC6. Allow 1 clk tick.
IDC7. Increment the loop counter and select BENO/CNTR OVFLW.
IDC8. DC007 will clk 100 (HEX) times.
IDC9. Select the CPU clk and send XFER REQ to flag the CPU.
12. Wait for XFER REQ from the IDC.
13. Clear CSR F<2:0> and send XFER GRANT to clr the XFER REQ.
14. Repeat Steps 11 - 14 169 (HEX) times.
15. Read the CSR and check <21:20> = 11.
16. Clear IDC.
17. End test.
Errors:
ERROR 1 - The CRC/ECC chip failed when determining a correctable
error or ECC STAT <1:0> bits failed = 11.
Printout:
EXPECTED RECEIVED
ECC STAT<1:0> ECC STAT<1:0>
CSR <21:20> CSR <21:20>
Debug:
ERROR 1 -
The IDC microcode will execute an instrcution with FUNC <3:0>=0001.
These bits will be decoded and will assert UCLR ECC L. This signal
is the Clear line to the CRC/ECC chip and will clear the CRC/ECC
chip internal Pattern register.
A one is then loaded into the Pattern register while UENC CRC H = L
and UDATA FIELD H = H.
The IDC microcode will then execute a series of instructions that will
assert ERR internal to the CRC/ECC chip.
In the next series of IDC micro-instructions, SYS CLK is selected which
is the W CLK input to the CRC/ECC chip. This clock will be used as the
internal clock for the CRC/ECC chip. Then UMAINT H is asserted and
UENB CRC H is deasserted. This will allow the internal Pattern register
to clock once each time this instruction is executed. This series of
micro-instructions is repeated until the internal conditions for a
correctable ECC error are simulated. This will set ECC status lines =
ECC STAT 1 H = H and ECC STAT 0 H = H which indicates a correctable
error has been found. The CPU will then read these lines as
CSR <21:20>.
--

```

```

:24960 T.2D:
U 15E4, B65E,15 :24961 MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT 15 IN WR[0] FOR
:24962 ;CONTROL AND STATUS WORD
U 15E5, 3E80,15 :24963 MOV WR[0] TO LS[CONTRL.STATUS] ;SET BIT 15 IN CONTROL AND
:24964 ;STATUS WORD - BIT 15
:24965 ;INDICATES START OF TEST TO
:24966 ;CONSOLE
U 15E6, 10E0,15 :24967 MISC [SET.CP.ATTN] ;ISSUE CPU ATTN TO CONSOLE
:24968 WAIT.T2D.0:
U 15E7, 095E,74 :24969 JMP [WAIT.T2D.0] ;LOOP HERE WAITING FOR
:24970 ;CONSOLE TO RESPOND
:24971
U 15E8, 65A0,15 :24972 CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
:24973 ;IN LS
:24974
U 15E9, B64A,15 :24975 MOV LS[IDC] TO WR[0] ;STORE MODULE CODE IN LS TO
U 15EA, 3E8C,15 :24976 MOV WR[0] TO LS[MODULE.NUM] ;INDICATE IDC MODULE
:24977
:24978
U 15EB, B640,15 :24979 MOV LS[#1] TO WR[0] ;SET WRO = 1
U 15EC, BE82,15 :24980 MOV WR[0] TO LS[ERROR.NUMBER] ;SET UP ERROR NUMBER = 1 IN LS
:24981
:24982
U 15ED, 37A4,15 :24983 MOV LS[#FFCFFFFFF] TO WR[0] ;MASK TO CHECK BITS <21:20>
U 15EE, 3E8A,15 :24984 MOV WR[0] TO LS[ERROR.MASK]
:24985
:24986
:24987
:24988 ;FORCE THE DC007 TO FIND A CORRECTABLE ERROR...ECC STAT<1:0>=11
:24989 ;CLR THE PATTERN REGISTER
:24990
U 15EF, B78E,15 :24991 MOV LS[DRIVE.STATUS] TO WR[0] ;HAS THE DRIVE SIZING ROUTINE
:24992 BIT LS[BIT7] WITH WR[0], ;BEEN RUN?
:24993 DT(LONG)&SET.ALU.CC
U 15F0, D94E,35 :24994 JMP.IF[BIT.SET] TO [SIZE.OK.T2D.1] ;IF YES THEN GO DIRCETLY TO
:24995 ;TEST
:24996
U 15F2, 8870,0C :24997 JSR [DRIVE.SIZE] ;GO RUN THE SIZING ROUTINE
:24998
:24999 SIZE.OK.T2D.1:
:25000
:25001 ;SELECT AN RL02 DRIVE
:25002
U 15F3, B78E,15 :25003 MOV LS[DRIVE.STATUS] TO WR[0] ;IS DRIVE 0 AN R80?
:25004 BIT LS[BIT0] WITH WR[0],
:25005 DT(LONG)&SET.ALU.CC
U 15F4, 5940,35 :25006 JMP.IF[BIT.CLR] TO [SET.DS0.T2D.1] ;JMP IF DRV 0 IS NOT AN R80
U 15F5, 095F,89 :25007 MOV LS[SETCSR.DS1] TO WR[2] ;SAVE THE DATA TO SEL DRIVE 1
:25008 ;IN WR[2]
U 15F7, 095F,94 :25009 JMP [TEST.T2D.1]
:25010 SET.DS0.T2D.1:
U 15F8, 3715,15 :25011 MOV LS[SETCSR.DS0] TO WR[2] ;SAVE THE DATA TO SEL DRIVE 0
:25012 ;IN WR[2]
:25013
:25014 TEST.T2D.1:
U 15F9, 8872,EC :25014 JSR [DIAG.CODE] ;FORCE THE IDC TO DIAG.IDLE

```



ZZ-ENKCF-1.4 :  
: ENKCF.MCR  
: ENKCF.MIC

ENKCF.MIC

MICRO2 1M(01)

Test 2D CORRECTABLE 15-MAR-83

15-MAR-83 11:46:22

ENKCF Micro-diagnostic for IDC module

Sequence 512

Page 506

Test 2D CORRECTABLE ECC ERROR TEST (M8388 IDC MODULE)

```

:25015 LOOP.T2D.1:
U 15FA, 17D8,15 :25016 PORT.CONTROL [SEL.FIFO.A] ;SELECT FIFO A
U 15FB, 17C0,15 :25017 PORT.CONTROL [CLEAR.FIFO.CNTR] ;SET ADRS = 0
U 15FC, B640,15 :25018 MOV LS[#1] TO WR[0] ;GET THE DATA TO WRITE
U 15FD, 97A8,15 :25019 PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;WRITE A ONE TO FIFO A
U 15FE, 17C0,15 :25020 PORT.CONTROL [CLEAR.FIFO.CNTR] ;SET ADRS = 0
:25021
U 15FF, 3712,95 :25022 MOV LS[SETCSR.CRDY] TO WR[1] ;SET UP THE CSR TO BRANCH TO
U 1600, 370E,15 :25023 MOV LS[SETCSR.F7] TO WR[0] ; THE CORRECT IDC ROUTINE
U 1601, AEC2,15 :25024 BIS WR[1] TO WR[0]
U 1602, 17A0,15 :25025 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:25026 ;MAINT=0,CRDY=1,F<2:0>=111
:25027 ;CONTINUE BRANCHING
:25028
U 1603, 379E,15 :25029 MOV LS[SETCSR.R80] TO WR[0] ;CONTINUE TO BRANCH TO ROUTINE
U 1604, 3700,95 :25030 MOV LS[SETCSR.F0] TO WR[1]
U 1605, AEC2,15 :25031 BIS WR[1] TO WR[0]
U 1606, 3712,95 :25032 MOV LS[SETCSR.CRDY] TO WR[1] ;SET CRDY TO HOLD SDIO LOW
U 1607, AEC2,15 :25033 BIS WR[1] TO WR[0]
U 1608, 17A0,15 :25034 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:25035 ;R80=1, F<2:0>=000
:25036 -----
:25037 -----
:25038 *****
:25039 * IDC *
:25040 *****
:25041 -----
:25042 :018:
:25043 :=000
:25044 :DIAG.IDLE:
:25045
:25046 BEN0/F0,
:25047 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:25048 BEN2/F2,
:25049 NAD/DIAG.IDLE
:25050 -----
:25051 :01F:
:25052 :=111
:25053
:25054 BEN0/CRDY.L,
:25055 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:25056 NAD/TEST.FUNC7 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:25057 -----
:25058 :078:
:25059 :=0*0
:25060 :TEST.FUNC7:
:25061
:25062 NAD/GROUP0
:25063 -----
:25064 :163:
:25065 :=0**
:25066 :GROUP0:
:25067
:25068 BEN2/R80.FORMAT.H, ;WHEN THE R80 FORMAT BIT SETS..BRANCH
:25069 NAD/GROUP0 ; TO NEXT WORD AND THEN BRANCH ON THE
```

ZZ-ENKCF-1.4 :  
: ENKCF.MCR  
: ENKCF.MIC

ENKCF.MIC

MICRO2 1M(01)

Test 2D CORRECTABLE 15-MAR-83

15-MAR-83 11:46:22

K 8

Fiche 3 Frame K8

Sequence 513

Rev 01.40

Page 507

Test 2D CORRECTABLE ECC ERROR TEST (M8388 IDC MODULE)

ZZ-1  
:  
:

```
25070 : ; FUNCTION BITS
25071 : -----
25072 : 167:
25073 : =1**
25074 :
25075 : BEN0/F0,
25076 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
25077 : BEN2/F2,
25078 : NAD/GROUP0.F0
25079 : -----
25080 : 020:
25081 : =000
25082 : GROUP0.F0:
25083 :
25084 : PAT.REG.1:
25085 :
25086 : FUNC/CLR.ECC, ;CLR ECC TO ZERO THE PATTERN REGISTER
25087 : NAD/PAT.REG.2
25088 : -----
25089 : 1E6:
25090 : PAT.REG.2:
25091 :
25092 : UCMD/SET.FIFOA, ;SELECT FIFO A
25093 : NAD/PAT.REG.3
25094 : -----
25095 : 1E7:
25096 : PAT.REG.3:
25097 :
25098 : ENB.FIFO/ON, ;LOAD THE DSR WITH DATA FROM THE FIFO
25099 : LOAD.DSR/ON,
25100 : UMAINT/ON, ;ASSERT MAINT SO DSR0 BECOMES NRZ DATA
25101 : NAD/PAT.REG.4
25102 : -----
25103 : 1E8:
25104 : PAT.REG.4:
25105 :
25106 : DATA.FIELD/ON, ;ALLOW READ CLK
25107 : MODE/ECC, ;ECC MODE TO SHIFT BIT 31
25108 : NAD/XFER.REQ ;FLAG THE CPU
25109 : -----
25110 : 1D8:
25111 : XFER.REQ:
25112 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
25113 : NAD/DIAG.WAIT
25114 : -----
25115 : 1D7:
25116 : DIAG.WAIT:
25117 :
25118 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
25119 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
25120 : -----
25121 : 159:
25122 : =*0*
25123 : DIAG.WAIT1:
25124 :
```

```

:25125 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:25126 : ; COMMAND FROM THE CPU
:25127 : -----
:25128 : 15B:
:25129 : =*1*
:25130 : DIAG.WAIT2:
:25131 :
:25132 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:25133 : ; WAIT SOME MORE
:25134 : -----
:25135 : WAIT.IDC.T2D.1:
U 1609, DB00,1A :25136 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ FROM IDC
U 160A, 0960,C4 :25137 : JMP [IDC.DONE.T2D.1] ;XFER REQ SET
U 160B, 0960,94 :25138 : JMP [WAIT.IDC.T2D.1] ;WAIT SOME MORE
:25139 :
:25140 : IDC.DONE.T2D.1:
U 160C, B700,15 :25141 : MOV LS[SETCSR.F0] TO WR[0] ;CLEAR CSR F<2:0>
U 160D, 17A0,15 :25142 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
U 160E, 90FA,15 :25143 : MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
:25144 :
:25145 :
U 160F, 370E,15 :25146 : MOV LS[SETCSR.F7] TO WR[0] ;SET UP THE CSR TO BRANCH TO
:25147 : ; THE CORRECT IDC ROUTINE
U 1610, 17A0,15 :25148 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:25149 : ;MAINT=0,CRDY=0,F<2:0>=111
:25150 :
:25151 : ;CONTINUE BRANCHING
:25152 :
U 1611, B79E,95 :25153 : MOV LS[SETCSR.R80] TO WR[1] ;CONTINUE TO BRANCH TO THE
U 1612, 3702,15 :25154 : MOV LS[SETCSR.F1] TO WR[0] ; IDC ROUTINE
U 1613, AEC2,15 :25155 : BIS WR[1] TO WR[0]
U 1614, 3712,95 :25156 : MOV LS[SETCSR.CRDY] TO WR[1]
U 1615, AEC2,15 :25157 : BIS WR[1] TO WR[0] ;SET CRDY TO HOLD SDIO LO
U 1616, 17A0,15 :25158 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:25159 : ;R80=1,F<2:0>=001
:25160 : -----
:25161 : -----
:25162 : *****
:25163 : * IDC *
:25164 : *****
:25165 : -----
:25166 : 018:
:25167 : =000
:25168 : DIAG.IDLE:
:25169 :
:25170 : BEN0/F0,
:25171 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:25172 : BEN2/F2,
:25173 : NAD/DIAG.IDLE
:25174 : -----
:25175 : 01F:
:25176 : =111
:25177 :
:25178 : BEN0/CRDY.L,
:25179 : BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE

```

```

:25180 : NAD/TEST.FUNC7 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:25181 : -----
:25182 : 078:
:25183 : =0*0
:25184 : TEST.FUNC7:
:25185 :
:25186 : NAD/GROUP0
:25187 : -----
:25188 : 163:
:25189 : =0**
:25190 : GROUP0:
:25191 :
:25192 : BEN2/R80.FORMAT.H, ; WHEN THE R80 FORMAT BIT SETS..BRANCH
:25193 : NAD/GROUP0 ; TO NEXT WORD AND THEN BRANCH ON THE
:25194 : ; FUNCTION BITS
:25195 : -----
:25196 : 167:
:25197 : =1**
:25198 :
:25199 : BEN0/F0,
:25200 : BEN1/F1, ; BRANCH ON THE CSR F<2:0> BITS
:25201 : BEN2/F2,
:25202 : NAD/GROUP0.F0
:25203 : -----
:25204 : 021:
:25205 : =001
:25206 : POS.REG.1:
:25207 :
:25208 : UCMD/SET.FIFOA, ; SEL FIFO A
:25209 : NAD/POS.REG.2
:25210 : -----
:25211 : 1E9:
:25212 : POS.REG.2:
:25213 :
:25214 : ENB.FIFO/ON, ; LOAD THE DSR WITH DATA FROM THE FIFO
:25215 : LOAD.DSR/ON,
:25216 : NAD/POS.REG.3
:25217 : -----
:25218 : 1EA:
:25219 : POS.REG.3:
:25220 :
:25221 : UMAINT/ON, ; ENABLE DSRO AS NRZ DATA
:25222 : CHECK.FIELD/ON, ; MAKE DCO07 BELIEVE IT HAS READ THE
:25223 : MODE/CRC, ; CHECK FIELD
:25224 : NAD/POS.REG.4
:25225 : -----
:25226 : 1EB:
:25227 : POS.REG.4:
:25228 :
:25229 : MODE/CRC, ; DROP CHECK FIELD WHILE THE INTERNAL
:25230 : ; DCO07 SHIFT REG IS NON-ZERO TO
:25231 : ; ASSERT ERROR INTERNAL TO THE DCO07
:25232 : ; ERROR ALLOWS THE POS REG TO INCR
:25233 : DATA.FIELD/ON, ; ASSERT DATA FIELD TO ALLOW CLK
:25234 : UMAINT/ON, ; TO SELECT DSRO AS NRZ DATA

```

```

:25235 : NAD/POS.REG.5
:25236 :-----
:25237 :1EC:
:25238 :POS.REG.5:
:25239 :
:25240 : DATA.FIELD/ON, ;TO ALLOW THE POS REG TO CLK ONCE
:25241 : MODE/CRC, ;TO PREVENT INTERNAL DCO07 SIGNALS
:25242 : UMAINT/ON, ;TO SEL DSRO AS NRZ DATA
:25243 : NAD/XFER.REQ ;FLAG THE CPU
:25244 :-----
:25245 :1D8:
:25246 :XFER.REQ:
:25247 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:25248 : NAD/DIAG.WAIT
:25249 :-----
:25250 :1D7:
:25251 :DIAG.WAIT:
:25252 :
:25253 : BEN1/XFER.RFQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:25254 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:25255 :-----
:25256 :159:
:25257 :=*0*
:25258 :DIAG.WAIT1:
:25259 :
:25260 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:25261 : ; COMMAND FROM THE CPU
:25262 :-----
:25263 :15B:
:25264 :=*1*
:25265 :DIAG.WAIT2:
:25266 :
:25267 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:25268 : ; WAIT SOME MORE
:25269 :-----
:25270 :WAIT.IDC.T2D.2:
:25271 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ FROM IDC
:25272 : JMP [IDC.DONE.T2D.2] ;XFER REQ SET
:25273 : JMP [WAIT.IDC.T2D.2] ;WAIT SOME MORE
:25274 :
:25275 :IDC.DONE.T2D.2:
:25276 : MOV LS[SETCSR.F0] TO WR[0] ;CLEAR CSR F<2:0>
:25277 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:25278 : MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
:25279 :
:25280 :
:25281 : MOV LS[#169] TO WR[3] ;USE AS A COUNTER
:25282 :
:25283 :LOOP.AGN.T2D:
:25284 :
:25285 : MOV LS[SETCSR.F7] TO WR[0] ;SET UP THE CSR TO BRANCH TO
:25286 : MOV LS[SETCSR.CRDY] TO WR[1] ; THE CORRECT IDC ROUTINE
:25287 : BIS WR[1] TO WR[0]
:25288 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:25289 : ;MAINT=0,CRDY=0,F<2:0>=111

```

U 1617, DB00,1A  
U 1618, 8961,A4  
U 1619, 0961,74

U 161A, B700,15  
U 161B, 17A0,15  
U 161C, 90FA,15

U 161D, B7B9,95

U 161E, 370E,15  
U 161F, 3712,95  
U 1620, AEC2,15  
U 1621, 17A0,15

U 16  
U 16  
U 16  
  
U 16  
U 16  
U 16  
  
U 16  
U 16  
U 16  
  
U 16  
U 16  
  
U 16  
U 16  
U 16  
U 16

```

:25290 ;CONTINUE BRANCHING
:25291
U 1622, B79E,95 :25292 MOV LS[SETCSR.R80] TO WR[1] ;CONTINUE TO BRANCH TO THE
U 1623, 370E,15 :25293 MOV LS[SETCSR.F7] TO WR[0] ; IDC ROUTINE
U 1624, AEC2,15 :25294 BIS WR[1] TO WR[0]
U 1625, 3712,95 :25295 MOV LS[SETCSR.CRDY] TO WR[1]
U 1626, AEC2,15 :25296 BIS WR[1] TO WR[0] ;SET CRDY TO HOLD SDIO LO
U 1627, AEC4,15 :25297 BIS WR[2] TO WR[0] ;SELECT THE RLO2 SYSTEM CLK
U 1628, 17A0,15 :25298 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:25299 ;R80=1,F<2:0>=111
:25300 -----
:25301 -----
:25302 *****
:25303 * IDC *
:25304 *****
:25305 -----
:25306 018:
:25307 =000
:25308 DIAG.IDLE:
:25309
:25310 BEN0/F0,
:25311 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:25312 BEN2/F2,
:25313 NAD/DIAG.IDLE
:25314 -----
:25315 01F:
:25316 =111
:25317
:25318 BEN0/CRDY.L,
:25319 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:25320 NAD/TEST.FUNC7 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:25321 -----
:25322 078:
:25323 =0*0
:25324 TEST.FUNC7:
:25325
:25326 NAD/GROUP0
:25327 -----
:25328 163:
:25329 =0**
:25330 GROUP0:
:25331
:25332 BEN2/R80.FORMAT.H, ;WHEN THE R80 FORMAT BIT SETS..BRANCH
:25333 NAD/GROUP0 ; TO NEXT WORD AND THEN BRANCH ON THE
:25334 ; FUNCTION BITS
:25335 -----
:25336 167:
:25337 =1**
:25338
:25339 BEN0/F0,
:25340 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:25341 BEN2/F2,
:25342 NAD/GROUP0.F0
:25343 -----
:25344 027:

```

```

25345 : =111
25346 : ECC.STAT.1:
25347 :
25348 : ENB.SYS.CLK, ;SELECT THE RL02 SYSTEM CLOCK
25349 : NAD/ECC.STAT.2
25350 : -----
25351 : 1F1:
25352 : ECC.STAT.2:
25353 :
25354 : LOAD.LOOP.CNTR.[00], ;LOAD THE LPCNTR = 00
25355 : NAD/ECC.STAT.3
25356 : -----
25357 : 196:
25358 : =0
25359 : ECC.STAT.3:
25360 :
25361 : INCR.LPCNTR/ON, ;INCR THE LOOP CNTR
25362 : UMAINT/ON, ;SELECS DSRO AS NRZ DATA
25363 : MODE/ECC, ;SELECT ECC MODE
25364 : BEN0/CNTR.OVFLW, ;LOOP HERE 255 TIMES
25365 : NAD/ECC.STAT.3
25366 : -----
25367 : 197:
25368 : =1
25369 : ECC.STAT.4:
25370 :
25371 : ENB.CPU.CLK, ;ENABLE THE CPU CLK
25372 : NAD/XFER.REQ ;FLAG THE CPU
25373 : -----
25374 : 1D8:
25375 : XFER.REQ:
25376 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
25377 : NAD/DIAG.WAIT
25378 : -----
25379 : 1D7:
25380 : DIAG.WAIT:
25381 :
25382 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
25383 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
25384 : -----
25385 : 159:
25386 : =*0*
25387 : DIAG.WAIT1:
25388 :
25389 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
25390 : ; COMMAND FROM THE CPU
25391 : -----
25392 : 15B:
25393 : =*1*
25394 : DIAG.WAIT2:
25395 :
25396 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
25397 : ; WAIT SOME MORE
25398 : -----
25399 : WAIT.IDC.T2D.3:

```

|              |   |           |                                                       |                    |                                       |              |
|--------------|---|-----------|-------------------------------------------------------|--------------------|---------------------------------------|--------------|
| ZZ-ENKCF-1.4 | : | ENKCF.MIC | Test 2D CORRECTABLE                                   | D 9<br>15-MAR-83   | Fiche 3 Frame D9                      | Sequence 519 |
| :            | : | ENKCF.MCR | MICRO2 1M(01)                                         | 15-MAR-83 11:46:22 | ENKCF Micro-diagnostic for IDC module | Rev 01.40    |
| :            | : | ENKCF.MIC | Test 2D CORRECTABLE ECC ERROR TEST (M8388 IDC MODULE) |                    |                                       | Page 513     |

  

|                 |   |       |                                      |   |                                |
|-----------------|---|-------|--------------------------------------|---|--------------------------------|
| U 1629, DB00,1A | : | 25400 | SKIP IF[NO.FAST.INTERRUPT]           | : | WAIT FOR XFER REQ FROM IDC     |
| U 162A, 8962,C4 | : | 25401 | JMP [IDC.DONE.T2D.3]                 | : | XFER REQ SET                   |
| U 162B, 8962,94 | : | 25402 | JMP [WAIT.IDC.T2D.3]                 | : | WAIT SOME MORE                 |
|                 | : | 25403 |                                      |   |                                |
|                 | : | 25404 | IDC.DONE.T2D.3:                      |   |                                |
| U 162C, B700,15 | : | 25405 | MOV LS[SETCSR.F0] TO WR[0]           | : | CLEAR CSR F<2:0>               |
| U 162D, 17A0,15 | : | 25406 | PORT.WRITE PORT.ADRS[CSR] WITH WR[0] | : | WRITE THE CSR                  |
| U 162E, 90FA,15 | : | 25407 | MISC2 [TRANSFER.GRANT]               | : | SEND GRANT TO CLR THE XFER REQ |
|                 | : | 25408 |                                      |   |                                |
|                 | : | 25409 | DEC WR[3],                           | : | DEC THE LOOP CNTR              |
| U 162F, A107,B5 | : | 25410 | DT(LONG)&SET.ALU.CC                  | : | DONE?                          |
| U 1630, 0961,E1 | : | 25411 | JMP IF[NEQ] TO [LOOP.AGN.T2D]        | : | JMP IF NOT DONE                |
|                 | : | 25412 |                                      |   |                                |
|                 | : | 25413 | READ.CSR.T2D.1:                      |   |                                |
| U 1631, 9090,15 | : | 25414 | MISC [SET.PORT.I.0]                  | : | SELECT THE IDC                 |
| U 1632, 9780,15 | : | 25415 | PORT.READ PORT.ADRS[CSR]             | : | SET UP TO READ THE CSR         |
| U 1633, 3B32,15 | : | 25416 | MOV PORT TO LS[PORT0]                | : | READ THE DATA FROM CSR AND     |
|                 | : | 25417 |                                      | : | SAVE IT IN LS                  |
| U 1634, 3732,15 | : | 25418 | MOV LS[PORT0] TO WR[0]               | : | SET UP RECEIVED DATA           |
| U 1635, 3668,95 | : | 25419 | MOV LS[BIT20] TO WR[1]               | : | SET UP EXPECTED DATA           |
| U 1636, C76A,95 | : | 25420 | BIS LS[BIT21] TO WR[1]               |   |                                |
| U 1637, 0869,3C | : | 25421 | JSR [CHECK.RESULT]                   | : | CHECK THE DATA                 |
| U 1638, 095F,A4 | : | 25422 | JMP [LOOP.T2D.1]                     | : | LOOP HERE IF ERR & LOE IS SET  |
|                 | : | 25423 |                                      |   |                                |
| U 1639, 8875,3C | : | 25424 | JSR [CLEANUP]                        | : | CLEANUP THE CONDITIONS SET BY  |
|                 | : | 25425 |                                      | : | THIS TEST                      |
|                 | : | 25426 | END.T2D:                             | : | END TEST                       |
|                 | : | 25427 |                                      |   |                                |
|                 | : | 25428 |                                      |   |                                |



:25429 .PAGE "Test 2E HEADER MATCH TEST (M8388 IDC MODULE)"

:25430 :++

:25431 :

:25432 :

:25433 :

:25434 :

:25435 :

:25436 :

:25437 :

:25438 :

:25439 :

:25440 :

:25441 :

:25442 :

:25443 :

:25444 :

:25445 :

:25446 :

:25447 :

:25448 :

:25449 :

:25450 :

:25451 :

:25452 :

:25453 :

:25454 :

:25455 :

:25456 :

:25457 :

:25458 :

:25459 :

:25460 :

:25461 :

:25462 :

:25463 :

:25464 :

:25465 :

:25466 :

:25467 :

:25468 :

:25469 :

:25470 :

:25471 :

:25472 :

:25473 :

:25474 :

:25475 :

:25476 :

:25477 :

:25478 :

:25479 :

:25480 :

:25481 :

:25482 :

:25483 :

Test Description:

This test is designed to insure that the mismatch logic recognizes a match between the read data and the serial DAR data. Because of the design of the diskless loop, DAR bit <9> will be compared with Bit<0> of the read data.  
Set the drive select to an RL02 drive or a line with no drive attached. The data pattern will be loaded into the FIFO and the DAR. The data will then be taken from the FIFO and shifted through the diskless loop as read data. The data also will be serialized through the DAR SERIALIZER PALS. The DAR data and the read data will then be compared in the DSR RD SEQ REG PAL. The IDC will then do a TEST MATCH with BEN2/MISMATCH L selected. If the MATCH was not detected, (MISMATCH L = L), then the DAR will be incremented ONCE. If a MATCH was detected then the DAR will be incremented twice. The CPU will then check the contents of the DAR.

Data Pattern:

| DAR  | READ DATA |
|------|-----------|
| 0200 | 01        |

Assumptions:

It is assumed that the diskless loop is fully operational.

Test Steps:

1. Set up error mask, error number and module number in LS (for error printouts) and clear the previous error number in LS.
2. Force the IDC microcode to DIAG.IDLE.
3. If the drive sizing routine has not been run previously, JSR to DRIVE.SIZE.
4. Select an RL02 drive.
5. Clear Match.
6. Load the data pattern into the DAR.
7. Load the data pattern into the FIFO.
8. Set up the CSR to branch to the correct IDC microcode routine.

\*\*\*\*\*  
\* IDC \*  
\*\*\*\*\*

- IDC1. Select the disk clock.  
IDC2. Shift out the sync character.  
IDC3. Follow with the data from the FIFO.  
IDC4. Assert UENB BC H so that the DAR will begin serializing after sync is seen.  
IDC5. Execute an instruction with UCON <1:0> = 10, TEST MATCH.  
IDC6. Select the CPU clk.  
IDC7. Select BEN2/MISMATCH L.  
IDC8. If MISMATCH L is asserted increment the DAR once.

U 1

U 1

U 1

U 1

```

25484 : If MISMATCH is not asserted (a MATCH was detected)
25485 : increment the DAR twice.
25486 : IDC9. Issue XFER.REQ to flag the CPU and go to DIAG.WAIT.
25487 :
25488 : 9. Wait for a XFER.REQ from IDC.
25489 : 10. Clear CSR <2:0> and issue XFER GRANT to clear the XFER.REQ.
25490 : 11. Check the DAR = 202 (DAR was loaded with 200 and should be
25491 : incremented twice).
25492 : 12. Clear Match.
25493 : 13. Clear IDC.
25494 : 14. End test.
25495 :
25496 : Errors:
25497 :
25498 : ERROR 1 - The mismatch logic failed when attempting to cause a match
25499 : between the serialized DAR and read data.
25500 :
25501 : Printout:
25502 : EXPECTED RECEIVED
25503 : DAR DAR
25504 :
25505 : Debug:
25506 : ERROR 1 -
25507 : The data pattern is loaded into the DAR (data = 200) and into FIFO A
25508 : address 0 (data = 01). A line with no disk drive attached or with an
25509 : RL02 drive attached is selected and will assert RL02 H which is inverted
25510 : to becomes R80 H = L. The IDC will branch to a routine that will load
25511 : the data from FIFO A address 0, into the DSR. This data is then
25512 : shifted through the RL02 diskless loop to become READ DATA H
25513 : (see RL02 DATA PATH TESTS if diskless loop is not functioning
25514 : correctly).
25515 : READ DATA H is an input to the SERIAL DAR LO PAL and it becomes
25516 : M.READ DATA H. This signal is an input to the DSK RD SEQ REG PAL.
25517 : As this data is being shifted through the diskless loop, the DAR data
25518 : is being serialized. The BIT COUNTER PAL is cleared when ENB CLR BC H
25519 : is asserted from the IDC microword when SYNC SEEN L.
25520 : R80 H will not be asserted at this time so the DAR will be serialized
25521 : in order starting with Bit <0> up to Bit <18>. For example:
25522 : when the bit counter = 2, bit 2 of the DAR will be output as MUX DAR H.
25523 : The serializing is done in the BIT COUNTER PAL and the SERIAL DAR LO
25524 : and SERIAL DAR HI PALS.
25525 : The DAR at this time = 200, so MUX DAR H should remain L until the
25526 : BIT COUNTER PAL = 01001 and then DAR bit <9> should be output as
25527 : MUX DAR H = H.
25528 : At the DSK RD SEQ REG PAL, SYNC SEEN L will be asserted at the next
25529 : CURRENT CLK H after SEL DSK CLK H = H and BYTE COMPARE H = H. After
25530 : SYNC SEEN L is asserted, COMPARE L will assert when SERIAL DAR H =
25531 : M.READ DATA H with UWRT CHK H = L. COMPARE L will assert on the first
25532 : match of SERIAL DAR H and M.READ DATA H and will remain asserted
25533 : unless a mismatch is encountered. Bits <8:0> of the DAR which are all
25534 : zeros will be compared to zeros coming in as M.READ DATA H. When bit
25535 : <9> of the DAR is selected Bit <0> of the data which was loaded into
25536 : FIFO A will be M.READ DATA H, so in this test COMPARE L should remain
25537 : asserted.
25538 : The IDC will then issue an instruction with UCON <1:0> = 10. These bits

```

:25539 : will be decoded by the DSK RD SEQ REG PAL to perform a TEST.MATCH.  
 :25540 : COMPARE L is asserted at this time, so MISMATCH L = H.  
 :25541 : The IDC microcode will then select BEN2/MISMATCH L (BEN2 S2 H = L,  
 :25542 : BEN2 S1 H = L, BEN2 S0 H = H). MISMATCH L will not be asserted so the  
 :25543 : microcode will branch to a series of instructions that will increment  
 :25544 : the DAR twice. Since the DAR was loaded with 200, the expected data  
 :25545 : is 202.  
 :25546 : --  
 :25547 : \*\*\*\*\*

|                 |        |                |                                                  |                                |
|-----------------|--------|----------------|--------------------------------------------------|--------------------------------|
| U 163A, B65E,15 | :25548 | T.2E:          | MOV LS[BEGIN.TEST] TO WR[0]                      | ;SET BIT 15 IN WR[0] FOR       |
|                 | :25549 |                |                                                  | ; CONTROL AND STATUS WORD      |
| U 163B, 3E80,15 | :25550 |                | MOV WR[0] TO LS[CONTROL.STATUS]                  | ;SET BIT 15 IN CONTROL AND     |
|                 | :25551 |                |                                                  | ; STATUS WORD - BIT 15         |
|                 | :25552 |                |                                                  | ; INDICATES START OF TEST TO   |
|                 | :25553 |                |                                                  | ; CONSOLE                      |
| U 163C, 10E0,15 | :25554 |                | MISC [SET.CP.ATTN]                               | ;ISSUE CPU ATTN TO CONSOLE     |
|                 | :25555 | WAIT.T2E.0:    |                                                  |                                |
| U 163D, 8963,D4 | :25556 |                | JMP [WAIT.T2E.0]                                 | ;LOOP HERE WAITING FOR         |
|                 | :25557 |                |                                                  | ; CONSOLE TO RESPOND           |
|                 | :25558 |                |                                                  |                                |
| U 163E, 65A0,15 | :25559 |                | CLR LS[PREVIOUS.ERROR]                           | ;CLEAR PREVIOUS ERROR NUMBER   |
|                 | :25560 |                |                                                  | ; IN LS                        |
|                 | :25561 |                |                                                  |                                |
| U 163F, B64A,15 | :25562 |                | MOV LS[IDC] TO WR[0]                             | ;STORE MODULE CODE IN LS TO    |
| U 1640, 3E8C,15 | :25563 |                | MOV WR[0] TO LS[MODULE.NUM]                      | ; INDICATE IDC MODULE          |
|                 | :25564 |                |                                                  |                                |
|                 | :25565 |                |                                                  |                                |
|                 | :25566 |                |                                                  |                                |
| U 1641, B640,15 | :25567 |                | MOV LS[#1] TO WR[0]                              | ;SET WR0 = 1                   |
| U 1642, BE82,15 | :25568 |                | MOV WR[0] TO LS[ERROR.NUMBER]                    | ;SET UP ERROR NUMBER = 1 IN LS |
|                 | :25569 |                |                                                  |                                |
| U 1643, 3776,15 | :25570 |                | MOV LS[DAR.MASK] TO WR[0]                        | ;GET THE MASK TO CHECK THE     |
| U 1644, 3E8A,15 | :25571 |                | MOV WR[0] TO LS[ERROR.MASK]                      | ; DAR                          |
|                 | :25572 |                |                                                  |                                |
| U 1645, B78E,15 | :25573 |                | MOV LS[DRIVE.STATUS] TO WR[0]                    | ;HAS THE DRIVE SIZING ROUTINE  |
|                 | :25574 |                | BIT LS[BIT7] WITH WR[0],                         | ; BEEN RUN?                    |
| U 1646, D94E,35 | :25575 |                | DT(LONG)&SET.ALU.CC                              |                                |
| U 1647, 8964,91 | :25576 |                | JMP.IF[BIT.SET] TO [SIZE.OK.T2E.1]               | ;IF YES THEN GO RUN TEST       |
|                 | :25577 |                |                                                  |                                |
| U 1648, 8870,0C | :25578 |                | JSR [DRIVE.SIZE]                                 | ;GO RUN THE SIZING ROUTINE     |
|                 | :25579 |                |                                                  |                                |
|                 | :25580 | SIZE.OK.T2E.1: |                                                  |                                |
| U 1649, 8872,EC | :25581 |                | JSR [DIAG.CODE]                                  | ;FORCE THE IDC TO DIAG.IDLE    |
|                 | :25582 |                |                                                  |                                |
|                 | :25583 |                | ;SELECT AN RL02 DRIVE AND SAVE THE INFO IN WR[2] |                                |
|                 | :25584 |                |                                                  |                                |
| U 164A, B78E,15 | :25585 |                | MOV LS[DRIVE.STATUS] TO WR[0]                    |                                |
|                 | :25586 |                | BIT LS[BIT0] WITH WR[0],                         | ;IS DRIVE 0 AN R80?            |
| U 164B, 5940,35 | :25587 |                | DT(LONG)&SET.ALU.CC                              |                                |
| U 164C, 0964,F9 | :25588 |                | JMP.IF[BITS.CLR] TO [SET.DS0.T2E.1]              | ;JMP IF DRV 0 IS NOT AN R80    |
| U 164D, B717,15 | :25589 |                | MOV LS[SETCSR.DS1] TO WR[2]                      | ;SAVE THE DATA TO SEL DRIVE 1  |
| U 164E, 0965,04 | :25590 |                | JMP [TEST.T2E.1]                                 |                                |
|                 | :25591 | SET.DS0.T2E.1: |                                                  |                                |
| U 164F, 3715,15 | :25592 |                | MOV LS[SETCSR.DS0] TO WR[2]                      | ;SAVE THE DATA TO SEL DRIVE 0  |
|                 | :25593 |                |                                                  |                                |

ZZ-ENKCF-1.4  
: ENKCF.MCR  
: ENKCF.MIC

ENKCF.MIC

Test 2E HEADER MAT15-MAR-83  
MICRO2 1M(01) 15-MAR-83 11:46:22  
Test 2E HEADER MATCH TEST (M8388 IDC MODULE)

Fiche 3 Frame H9

Sequence 523  
Rev 01.40 Page 517

ZZ-  
:  
:

```
:25594 TEST.T2E.1:
U 1650, 370E,15 :25595 MOV LS[SETCSR.F7] TO WR[0]
U 1651, 17A0,15 :25596 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:25597 ;MAINT=0,CRDY=0,F<2:0>=111
:25598 ;CONTINUE TO BRANCH
U 1652, 379E,15 :25599 MOV LS[SETCSR.R80] TO WR[0] ;SET UP THE CSR TO CONTINUE TO
U 1653, 3700,95 :25600 MOV LS[SETCSR.F0] TO WR[1] ; BRANCH TO THE CORRECT ROUTINE
U 1654, AEC2,15 :25601 BIS WR[1] TO WR[0]
U 1655, 17A0,15 :25602 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:25603 ;F<2:0> = 000
:25604 ;UCLR MATCH
:25605 -----
:25606 -----
:25607 *****
:25608 * IDC *
:25609 *****
:25610 -----
:25611 :018:
:25612 :=000
:25613 :DIAG.IDLE:
:25614
:25615 BEN0/F0,
:25616 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:25617 BEN2/F2,
:25618 NAD/DIAG.IDLE
:25619 -----
:25620 :01F:
:25621 :=111
:25622
:25623 BEN0/CRDY.L,
:25624 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:25625 NAD/TEST.FUNC7 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:25626 -----
:25627 :079:
:25628 :=0*1
:25629
:25630 NAD/GROUP1
:25631 -----
:25632 :182:
:25633 :=0**
:25634 :GROUP1:
:25635
:25636 BEN2/R80.FORMAT.H, ;WHEN THE R80 FORMAT BIT SETS..BRANCH
:25637 NAD/GROUP1 ; TO NEXT WORD AND THEN BRANCH ON THE
:25638 ; FUNCTION BITS
:25639 -----
:25640 :186:
:25641 :=1**
:25642
:25643 BEN0/F0,
:25644 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:25645 BEN2/F2,
:25646 NAD/GROUP1.F0
:25647 -----
:25648 :028:
```

ZZ-ENKCF-1.4 ;  
; ENKCF.MCR  
; ENKCF.MIC

ENKCF.MIC

MICRO2 1M(01)

Test 2E HEADER MAT15-MAR-83

15-MAR-83 11:46:22

Fiche 3 Frame I9

Sequence 524

Rev 01.40

Page 518

ZZ-1  
:  
:

Test 2E HEADER MATCH TEST (M8388 IDC MODULE)

```
:25649 : =000
:25650 : GROUP1.F0:
:25651 :
:25652 : CLR.MATCH.1:
:25653 :
:25654 : UCON/CLR.MATCH, ;CLR THE MATCH CONDITION
:25655 : NAD/TEST.MATCH.1
:25656 : -----
:25657 : 1F6:
:25658 : TEST.MATCH.1:
:25659 :
:25660 : BEN2/MISMATCH.L, ;IS MATCH ASSERTED?
:25661 : NAD/BEN2.1
:25662 : -----
:25663 : 1A0:
:25664 : =0**
:25665 : BEN2.1:
:25666 :
:25667 : INCR.DAR/ON, ;INCR DAR
:25668 : NAD/XFER.REQ ;FLAG THE CPU
:25669 : -----
:25670 : 1A4:
:25671 : =1**
:25672 : BEN2.2:
:25673 :
:25674 : INCR.DAR/ON, ;INCR DAR IF BEN2 CONDITION ASSERTED
:25675 : NAD/BEN2.1 ;INCR DAR AGAIN IF BEN2 COND ASSERTED
:25676 : -----
:25677 : 1D8:
:25678 : XFER.REQ:
:25679 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:25680 : NAD/DIAG.WAIT
:25681 : -----
:25682 : 1D7:
:25683 : DIAG.WAIT:
:25684 :
:25685 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:25686 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:25687 : -----
:25688 : 159:
:25689 : =*0*
:25690 : DIAG.WAIT1:
:25691 :
:25692 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:25693 : ; COMMAND FROM THE CPU
:25694 : -----
:25695 : 15B:
:25696 : =*1*
:25697 : DIAG.WAIT2:
:25698 :
:25699 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:25700 : ; WAIT SOME MORE
:25701 : -----
:25702 : WAIT.IDC.T2E.1:
:25703 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ FROM IDC
```

U 1656, DB00,1A

[illegible]

```

:25759 : =0*1
:25760 :
:25761 : NAD/GROUP1
:25762 : -----
:25763 : 182:
:25764 : =0**
:25765 : GROUP1:
:25766 :
:25767 : BEN2/R80.FORMAT.H, ; WHEN THE R80 FORMAT BIT SETS..BRANCH
:25768 : NAD/GROUP1 ; TO NEXT WORD AND THEN BRANCH ON THE
:25769 : ; FUNCTION BITS
:25770 : -----
:25771 : 186:
:25772 : =1**
:25773 :
:25774 : BEN0/F0,
:25775 : BEN1/F1, ; BRANCH ON THE CSR F<2:0> BITS
:25776 : BEN2/F2,
:25777 : NAD/GROUP1.F0
:25778 : -----
:25779 : 02A:
:25780 : =010
:25781 : MATCH.OR.WRTCHK.1:
:25782 :
:25783 : UMAINT/ON,
:25784 : LOOP.LOCK/ON, ; MAINT PREVENTS LOOP LOCK FROM SELECTING
:25785 : DISK.WRITE/ON, ; READ DATA...DISK WRITE STILL SELECTS DSRO
:25786 : LOAD.LOOP.CNTR.[0E0],
:25787 : NAD/MATCH.OR.WRTCHK.2
:25788 : -----
:25789 : 198:
:25790 : =0
:25791 : MATCH.OR.WRTCHK.2:
:25792 :
:25793 : UMAINT/ON,
:25794 : LOOP.LOCK/ON, ; SHIFT OUT 32 ZEROS TO SYNC UP
:25795 : DISK.WRITE/ON, ; DATA SEPARATOR
:25796 : UCMD/SET.FIFOA,
:25797 : INCR.LPCNTR/ON,
:25798 : BEN0/CNTR.OVFLW,
:25799 : NAD/MATCH.OR.WRTCHK.2
:25800 : -----
:25801 : 199:
:25802 : =1
:25803 : MATCH.OR.WRTCHK.3:
:25804 :
:25805 : UMAINT/ON,
:25806 : LOOP.LOCK/ON,
:25807 : DISK.WRITE/ON, ; MUST WAIT 3 CYCLES FOR SYNC SEEN
:25808 : ENB.DSK.CLK,
:25809 : ENB.CLR.BC/ON, ; CLR THE BIT COUNTER WHEN SYNC SEEN
:25810 : NAD/MATCH.OR.WRTCHK.4 ; FOR THE MATCH TEST
:25811 : -----
:25812 : 1F2:
:25813 : MATCH.OR.WRTCHK.4:

```

```

25814 : UMAINT/ON,
25815 : LOOP.LOCK/ON, ;WAIT SOME MORE
25816 : DISK.WRITE/ON,
25817 : ENB.CLR.BC/ON, ;CLR THE BIT COUNTER WHEN SYNC SEEN
25818 : NAD/MATCH.OR.WRTCHK.5 ; FOR THE MATCH TEST
25819 :-----
25820 :1F3:
25821 :MATCH.OR.WRTCHK.5:
25822 :
25823 : UMAINT/ON,
25824 : LOOP.LOCK/ON, ;WAIT SOME MORE
25825 : DISK.WRITE/ON,
25826 : ENB.CLR.BC/ON, ;CLR THE BIT COUNTER WHEN SYNC SEEN
25827 : NAD/MATCH.OR.WRTCHK.6 ; FOR THE MATCH TEST
25828 :-----
25829 :1F4:
25830 :MATCH.OR.WRTCHK.6:
25831 :
25832 : UMAINT/ON,
25833 : LOOP.LOCK/ON,
25834 : DISK.WRITE/ON, ;LOAD THE DATA FROM FIFO A INTO THE DSR
25835 : ENB.FIFO/ON,
25836 : LOAD.DSR/ON,
25837 : BEN2/MAINT, ;MATCH TEST OR WRTCHK TEST?
25838 : NAD/MATCH.OR.WRTCHK.7
25839 :-----
25840 :183:
25841 :=0**
25842 :MATCH.OR.WRTCHK.7:
25843 :
25844 : UMAINT/ON,
25845 : LOOP.LOCK/ON, ;MATCH TEST
25846 : DISK.WRITE/ON,
25847 : LOAD.LOOP.CNTR.[OF6],
25848 : NAD/MATCH.1
25849 :-----
25850 :19A:
25851 :=0
25852 :MATCH.1:
25853 :
25854 : UMAINT/ON,
25855 : LOOP.LOCK/ON,
25856 : DISK.WRITE/ON, ;WAIT FOR DATA TO SHIFT THRU DISKLESS
25857 : INCR.LPCNTR/ON, ; LOOP TO BECOME RD DATA
25858 : BEN0/CNTR.OVFLW,
25859 : NAD/MATCH.1
25860 :-----
25861 :19B:
25862 :=1
25863 :MATCH.2:
25864 :
25865 : UMAINT/ON,
25866 : LOOP.LOCK/ON, ;KEEP THE CLK FROM THE DATA SEPARATOR
25867 : DISK.WRITE/ON, ; GOING AND TEST MATCH
25868 :

```

[illegible]



ZZ-ENKCF-1.4  
: ENKCF.MCR  
: ENKCF.MIC

ENKCF.MIC

MICRO2 1M(01)

Test 2E HEADER MAT15-MAR-83

15-MAR-83 11:46:22

Test 2E HEADER MATCH TEST (M8388 IDC MODULE)

M 9

Fiche 3 Frame M9

Sequence 528

Rev 01.40

Page 522

ZZ-1

:

:

```
:25869 : UCON/TEST.MATCH,
:25870 : NAD/MATCH.3
:25871 :-----
:25872 :1F5:
:25873 :MATCH.3:
:25874 :
:25875 : UMAINT/ON, ;KEEP THE CLK FROM THE DATA SEPARATOR
:25876 : LOOP.LOCK/ON, ; GOING
:25877 : DISK.WRITE/ON,
:25878 : ENB.CPU.CLK, ;SELECT THE CPU CLK
:25879 : NAD/TEST.MATCH.1
:25880 :-----
:25881 :1F6:
:25882 :TEST.MATCH.1:
:25883 :
:25884 : BEN2/MISMATCH.L, ;IS MATCH ASSERTED?
:25885 : NAD/BEN2.1
:25886 :-----
:25887 :1A0:
:25888 :=*0*
:25889 :BEN2.1:
:25890 :
:25891 : INCR.DAR/ON, ;INCR DAR
:25892 : NAD/XFER.REQ ;FLAG THE CPU
:25893 :-----
:25894 :1A4:
:25895 :=*1*
:25896 :BEN2.2:
:25897 :
:25898 : INCR.DAR/ON, ;INCR DAR IF BEN2 CONDITION ASSERTED
:25899 : NAD/BEN2.1 ;INCR DAR AGAIN IF BEN2 COND ASSERTED
:25900 :-----
:25901 :1D8:
:25902 :XFER.REQ:
:25903 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:25904 : NAD/DIAG.WAIT
:25905 :-----
:25906 :1D7:
:25907 :DIAG.WAIT:
:25908 :
:25909 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:25910 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:25911 :-----
:25912 :159:
:25913 :=*0*
:25914 :DIAG.WAIT1:
:25915 :
:25916 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:25917 : ; COMMAND FROM THE CPU
:25918 :-----
:25919 :15B:
:25920 :=*1*
:25921 :DIAG.WAIT2:
:25922 :
:25923 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
```

```

ZZ-ENKCF-1.4 : ENKCF.MIC Test 2E HEADER MAT15-MAR-83 Fiche 3 Frame N9 Sequence 529 Page 523 ZZ-E
: ENKCF.MCR : MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40
: ENKCF.MIC : Test 2E HEADER MATCH TEST (M8388 IDC MODULE)
:
:25924 : : : WAIT SOME MORE
:25925 :-----:
:25926 : WAIT.IDC.T2E.2:
U 166A, DB00,1A :25927 : SKIP.IF[NO.FAST.INTERUPT] : WAIT FOR XFER REQ FROM IDC
U 166B, 8966,D4 :25928 : JMP [IDC.DONE.T2E.2] : XFER REQ SET
U 166C, 0966,A4 :25929 : JMP [WAIT.IDC.T2E.2] : WAIT SOME MORE
:25930
:25931 : IDC.DONE.T2E.2:
U 166D, B700,15 :25932 : MOV LS[SETCSR.F0] TO WR[0] : CLEAR CSR F<2:0>
U 166E, 17A0,15 :25933 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] : WRITE THE CSR
U 166F, 90FA,15 :25934 : MISC2 [TRANSFER.GRANT] : SEND GRANT TO CLR THE XFER REQ
:25935
:25936 : READ.DAR.T2E.1:
U 1670, 9090,15 :25937 : MISC [SET.PORT.I.0] : SELECT THE IDC
U 1671, 1784,15 :25938 : PORT.READ PORT.ADRS[DAR] : SET UP TO READ THE DAR
U 1672, 3B32,15 :25939 : MOV PORT TO LS[PORT0] : READ THE DATA FROM DAR AND
:25940 : : : SAVE IT IN LS
U 1673, 3732,15 :25941 : MOV LS[PORT0] TO WR[0] : SET UP RECEIVED DATA
U 1674, 3652,95 :25942 : MOV LS[BIT9] TO WR[1] : SET UP EXPECTED DATA
U 1675, C742,95 :25943 : BIS LS[BIT1] TO WR[1]
U 1676, 0869,3C :25944 : JSR [CHECK.RESULT] : CHECK THE DATA
U 1677, 0965,C4 :25945 : JMP [LOOP.T2E.1] : LOOP HERE IF ERR & LOE IS SET
:25946
:25947 : CLEANUP.T2E:
:25948 : CLR MATCH AS PART OF CLEANUP
:25949
U 1678, 370E,15 :25950 : MOV LS[SETCSR.F7] TO WR[0]
U 1679, 17A0,15 :25951 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] : WRITE THE CSR
:25952 : : : MAINT=0,CRDY=0,F<2:0>=111
:25953 : CONTINUE TO BRANCH
U 167A, 379E,15 :25954 : MOV LS[SETCSR.R80] TO WR[0] : SET UP THE CSR TO CONTINUE TO
U 167B, 3700,95 :25955 : MOV LS[SETCSR.F0] TO WR[1] : BRANCH TO THE CORRECT ROUTINE
U 167C, AEC2,15 :25956 : BIS WR[1] TO WR[0]
U 167D, 17A0,15 :25957 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] : WRITE THE CSR
:25958 : : : F<2:0> = 000
:25959 : UCLR MATCH
:25960 :-----:
:25961 :-----:
:25962 : : *****
:25963 : : * IDC *
:25964 : : *****
:25965 :-----:
:25966 : 018:
:25967 : =000
:25968 : DIAG.IDLE:
:25969
:25970 : BEN0/F0,
:25971 : BEN1/F1, : BRANCH ON THE CSR F<2:0> BITS
:25972 : BEN2/F2,
:25973 : NAD/DIAG.IDLE
:25974 :-----:
:25975 : 01F:
:25976 : =111
:25977
:25978 : BEN0/CRDY.L,

```

```

25979 : BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
25980 : NAD/TEST.FUNC7 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
25981 : -----
25982 : 079:
25983 : =0*1
25984 :
25985 : NAD/GROUP1
25986 : -----
25987 : 182:
25988 : =0**
25989 : GROUP1:
25990 :
25991 : BEN2/R80.FORMAT.H, ;WHEN THE R80 FORMAT BIT SETS..BRANCH
25992 : NAD/GROUP1 ; TO NEXT WORD AND THEN BRANCH ON THE
25993 : ; FUNCTION BITS
25994 : -----
25995 : 186:
25996 : =1**
25997 :
25998 : BEN0/F0,
25999 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
26000 : BEN2/F2,
26001 : NAD/GROUP1.F0
26002 : -----
26003 : 028:
26004 : =000
26005 : GROUP1.F0:
26006 :
26007 : CLR.MATCH.1:
26008 :
26009 : UCON/CLR.MATCH, ;CLR THE MATCH CONDITION
26010 : NAD/TEST.MATCH.1
26011 : -----
26012 : 1F6:
26013 : TEST.MATCH.1:
26014 :
26015 : BEN2/MISMATCH.L, ;IS MATCH ASSERTED?
26016 : NAD/BEN2.1
26017 : -----
26018 : 1A0:
26019 : =0**
26020 : BEN2.1:
26021 :
26022 : INCR.DAR/ON, ;INCR DAR
26023 : NAD/XFER.REQ ;FLAG THE CPU
26024 : -----
26025 : 1A4:
26026 : =1**
26027 : BEN2.2:
26028 :
26029 : INCR.DAR/ON, ;INCR DAR IF BEN2 CONDITION ASSERTED
26030 : NAD/BEN2.1 ;INCR DAR AGAIN IF BEN2 COND ASSERTED
26031 : -----
26032 : 1D8:
26033 : XFER.REQ:

```

U 1  
U 1  
  
U 1  
U 1  
  
U 1  
  
U 1  
U 1  
  
U 1  
U 1  
  
U 1  
U 1

111

111

22

119

101

# 1

1

1

101

01  
41

101

100

1

```

:26129 : 10. Clear CSR F<2:0> and send XFER GRANT to clear XFER REQ.
:26130 : 11. Set up the CSR to branch to the correct IDC routine.
:26131 :
:26132 :
:26133 : *****
:26134 : * IDC *
:26135 : *****
:26136 :
:26137 :
:26138 : IDC6. Execute an instruction with UCON <1:0> = 01, CLEAR MATCH.
:26139 : IDC7. Select BEN2/MISMATCH L.
:26140 : IDC8. If MISMATCH L is asserted (L), increment the DAR once.
:26141 : If MISMATCH L is not asserted (a MATCH was detected)
:26142 : then increment the DAR twice.
:26143 : IDC9. Issue XFER.REQ to flag the CPU.
:26144 :
:26145 : 12. Wait for a XFER.REQ from IDC.
:26146 : 13. Clear CSR F<2:0> and send XFER GRANT to clear XFER.REQ.
:26147 : 14. Check the DAR = 1.
:26148 : 15. Clear IDC.
:26149 : 16. End test.
:26150 :
:26151 : Errors:
:26152 :
:26153 : ERROR 1 - The BEN2/MISMATCH L branch failed or the CLEAR MATCH
:26154 : function did not work.
:26155 :
:26156 :
:26157 : Printout:
:26158 : EXPECTED RECEIVED
:26159 : DAR DAR
:26160 :
:26161 : Debug:
:26162 :
:26163 : ERROR 1 -
:26164 : A match condition is produced between SERIAL DAR H and M.READ DATA H
:26165 : (MISMATCH L = H). The DAR is then cleared.
:26166 : The IDC will then issue an instruction with UCON <1:0> = 01(CLR.MATCH).
:26167 : These bits will be decoded by the DSK RD SEQ REG PAL and will assert
:26168 : MISMATCH L. The IDC microcode will then select BEN2/MISMATCH L.
:26169 : Since MISMATCH L = L, NUA 2 H = L and the microcode will branch to
:26170 : an instruction that will increment the DAR once.
:26171 : --
:26172 : *****
:26173 : T.2F:
:26174 : MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT 15 IN WR[0] FOR
:26175 : MOV WR[0] TO LS[CONTROL.STATUS] ;CONTROL AND STATUS WORD
:26176 : ;SET BIT 15 IN CONTROL AND
:26177 : ;STATUS WORD - BIT 15
:26178 : ;INDICATES START OF TEST TO
:26179 : ;CONSOLE
:26180 : MISC [SET.CP.ATTN] ;ISSUE CPU ATTN TO CONSOLE
:26181 : WAIT.T2F.0:
:26182 : JMP [WAIT.T2F.0] ;LOOP HERE WAITING FOR
:26183 : ;CONSOLE TO RESPOND

```

U 1685, B65E,15  
 U 1686, 3E80,15  
 U 1687, 10E0,15  
 U 1688, 0968,84

[illegible]

```

U 16A7, AEC4,15 :26239 BIS WR[2] TO WR[0] ;INCLUDE THE RLO2 DRV SELECT
U 16A8, 17A0,15 :26240 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:26241 -----
:26242 -----
:26243 *****
:26244 * IDC *
:26245 *****
:26246 -----
:26247 018:
:26248 =000
:26249 DIAG.IDLE:
:26250
:26251 BEN0/F0,
:26252 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:26253 BEN2/F2,
:26254 NAD/DIAG.IDLE
:26255 -----
:26256 01F:
:26257 =111
:26258
:26259 BEN0/CRDY.L,
:26260 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:26261 NAD/TEST.FUNC7 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:26262 -----
:26263 079:
:26264 =0*1
:26265
:26266 NAD/GROUP1
:26267 -----
:26268 182:
:26269 =0**
:26270 GROUP1:
:26271
:26272 BEN2/R80.FORMAT.H, ;WHEN THE R80 FORMAT BIT SETS..BRANCH
:26273 NAD/GROUP1 ; TO NEXT WORD AND THEN BRANCH ON THE
:26274 ; FUNCTION BITS
:26275 -----
:26276 186:
:26277 =1**
:26278
:26279 BEN0/F0,
:26280 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:26281 BEN2/F2,
:26282 NAD/GROUP1.F0
:26283 -----
:26284 02A:
:26285 =010
:26286 MATCH.OR.WRTCHK.1:
:26287
:26288 UMAINT/ON,
:26289 LOOP.LOCK/ON, ;MAINT PREVENTS LOOP LOCK FROM SELECTING
:26290 DISK.WRITE/ON, ;READ DATA...DISK WRITE STILL SELECTS DSRO
:26291 LOAD.LOOP.CNTR.[0E0],
:26292 NAD/MATCH.OR.WRTCHK.2
:26293 -----

```



ZZ-ENKCF-1.4 :  
: ENKCF.MCR  
: ENKCF.MIC

ENKCF.MIC

MICRO2 1M(01)

Test 2F CLEAR MATC15-MAR-83

15-MAR-83 11:46:22

Test 2F CLEAR MATCH TEST (M8388 IDC MODULE)

H 10

Fiche 3 Frame H10

Sequence 536

Rev 01.40

Page 530

ZZ-E  
:  
:

```
:26294 :198:
:26295 :=0
:26296 :MATCH.OR.WRTCHK.2:
:26297 :
:26298 : UMAINT/ON,
:26299 : LOOP.LOCK/ON, ;SHIFT OUT 32 ZEROS TO SYNC UP
:26300 : DISK.WRITE/ON, ; DATA SEPARATOR
:26301 : UCMD/SET.FIFOA,
:26302 : INCR.LPCNTR/ON,
:26303 : BEN0/CNTR.OVFLW,
:26304 : NAD/MATCH.OR.WRTCHK.2
:26305 :-----
:26306 :199:
:26307 :=1
:26308 :MATCH.OR.WRTCHK.3:
:26309 :
:26310 : UMAINT/ON,
:26311 : LOOP.LOCK/ON,
:26312 : DISK.WRITE/ON, ;MUST WAIT 3 CYCLES FOR SYNC SEEN
:26313 : ENB.DSK.CLK,
:26314 : ENB.CLR.BC/ON, ;CLR THE BIT COUNTER WHEN SYNC SEEN
:26315 : NAD/MATCH.OR.WRTCHK.4 ; FOR THE MATCH TEST
:26316 :-----
:26317 :1F2:
:26318 :MATCH.OR.WRTCHK.4:
:26319 :
:26320 : UMAINT/ON,
:26321 : LOOP.LOCK/ON, ;WAIT SOME MORE
:26322 : DISK.WRITE/ON,
:26323 : ENB.CLR.BC/ON, ;CLR THE BIT COUNTER WHEN SYNC SEEN
:26324 : NAD/MATCH.OR.WRTCHK.5 ; FOR THE MATCH TEST
:26325 :-----
:26326 :1F3:
:26327 :MATCH.OR.WRTCHK.5:
:26328 :
:26329 : UMAINT/ON,
:26330 : LOOP.LOCK/ON, ;WAIT SOME MORE
:26331 : DISK.WRITE/ON,
:26332 : ENB.CLR.BC/ON, ;CLR THE BIT COUNTER WHEN SYNC SEEN
:26333 : NAD/MATCH.OR.WRTCHK.6 ; FOR THE MATCH TEST
:26334 :-----
:26335 :1F4:
:26336 :MATCH.OR.WRTCHK.6:
:26337 :
:26338 : UMAINT/ON,
:26339 : LOOP.LOCK/ON,
:26340 : DISK.WRITE/ON, ;LOAD THE DATA FROM FIFO A INTO THE DSR
:26341 : ENB.FIFO/ON,
:26342 : LOAD.DSR/ON,
:26343 : BEN2/MAINT, ;MATCH TEST OR WRTCHK TEST?
:26344 : NAD/MATCH.OR.WRTCHK.7
:26345 :-----
:26346 :183:
:26347 :=0**
:26348 :MATCH.OR.WRTCHK.7:
```

```

:26349 :
:26350 : UMAINT/ON,
:26351 : LOOP.LOCK/ON, ;MATCH TEST
:26352 : DISK.WRITE/ON,
:26353 : LOAD.LOOP.CNTR.[OF6],
:26354 : NAD/MATCH.1
:26355 :-----
:26356 :19A:
:26357 :=0
:26358 :MATCH.1:
:26359 :
:26360 : UMAINT/ON,
:26361 : LOOP.LOCK/ON,
:26362 : DISK.WRITE/ON, ;WAIT FOR DATA TO SHIFT THRU DISKLESS
:26363 : INCR.LPCNTR/ON, ; LOOP TO BECOME RD DATA
:26364 : BEN0/CNTR.OVFLW,
:26365 : NAD/MATCH.1
:26366 :-----
:26367 :19B:
:26368 :=1
:26369 :MATCH.2:
:26370 :
:26371 : UMAINT/ON, ;KEEP THE CLK FROM THE DATA SEPARATOR
:26372 : LOOP.LOCK/ON, ; GOING AND TEST MATCH
:26373 : DISK.WRITE/ON,
:26374 : UCON/TEST.MATCH,
:26375 : NAD/MATCH.3
:26376 :-----
:26377 :1F5:
:26378 :MATCH.3:
:26379 :
:26380 : UMAINT/ON, ;KEEP THE CLK FROM THE DATA SEPARATOR
:26381 : LOOP.LOCK/ON, ; GOING
:26382 : DISK.WRITE/ON,
:26383 : ENB.CPU.CLK, ;SELECT THE CPU CLK
:26384 : NAD/TEST.MATCH.1
:26385 :-----
:26386 :1F6:
:26387 :TEST.MATCH.1:
:26388 :
:26389 : BEN2/MISMATCH.L, ;IS MATCH ASSERTED?
:26390 : NAD/BEN2.1
:26391 :-----
:26392 :1A0:
:26393 :=0**
:26394 :BEN2.1:
:26395 :
:26396 : INCR.DAR/ON, ;INCR DAR
:26397 : NAD/XFER.REQ ;FLAG THE CPU
:26398 :-----
:26399 :1A4:
:26400 :=1**
:26401 :BEN2.2:
:26402 :
:26403 : INCR.DAR/ON, ;INCR DAR IF BEN2 CONDITION ASSERTED

```

[illegible]

```

:26459 :-----
:26460 : *****
:26461 : * IDC *
:26462 : *****
:26463 :-----
:26464 :018:
:26465 :=000
:26466 :DIAG.IDLE:
:26467 :
:26468 : BEN0/F0,
:26469 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:26470 : BEN2/F2,
:26471 : NAD/DIAG.IDLE
:26472 :-----
:26473 :01F:
:26474 :=111
:26475 :
:26476 : BEN0/CRDY.L,
:26477 : BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:26478 : NAD/TEST.FUNC7 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:26479 :-----
:26480 :079:
:26481 :=0*1
:26482 :
:26483 : NAD/GROUP1
:26484 :-----
:26485 :182:
:26486 :=0**
:26487 :GROUP1:
:26488 :
:26489 : BEN2/R80.FORMAT.H, ;WHEN THE R80 FORMAT BIT SETS..BRANCH
:26490 : NAD/GROUP1 ; TO NEXT WORD AND THEN BRANCH ON THE
:26491 : ; FUNCTION BITS
:26492 :-----
:26493 :186:
:26494 :=1**
:26495 :
:26496 : BEN0/F0,
:26497 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:26498 : BEN2/F2,
:26499 : NAD/GROUP1.F0
:26500 :-----
:26501 :028:
:26502 :=000
:26503 :GROUP1.F0:
:26504 :
:26505 :CLR.MATCH.1:
:26506 :
:26507 : UCON/CLR.MATCH, ;CLR THE MATCH CONDITION
:26508 : NAD/TEST.MATCH.1
:26509 :-----
:26510 :1F6:
:26511 :TEST.MATCH.1:
:26512 :
:26513 : BEN2/MISMATCH.L, ;IS MATCH ASSERTED?

```

|      |   |
|------|---|
| U    | 1 |
| U    | 1 |
| U    | 1 |
| <br> |   |
| U    | 1 |
| U    | 1 |
| U    | 1 |
| <br> |   |
| U    | 1 |
| U    | 1 |
| <br> |   |
| U    | 1 |
| U    | 1 |
| U    | 1 |
| U    | 1 |
| U    | 1 |

```

U 16C0, 3732,15 :26569
U 16C1, 3640,95 :26570
U 16C2, 0869,3C :26571
U 16C3, 096A,F4 :26572
:26573
:26574
U 16C4, 8875,3C :26575
:26576
:26577 END.T2F:
:26578
:26579

MOV LS[PORT0] TO WR[0]
MOV LS[#1] TO WR[1]
JSR [CHECK.RESULT]
JMP [LOOP.T2F.1]

: SAVE IT IN LS
: SET UP RECEIVED DATA
: SET UP EXPECTED DATA
: CHECK THE DATA
: LOOP HERE IF ERR & LOE IS SET

: CLEANUP THE CONDITIONS SET
: BY THIS TEST
: END TEST

```

ZZ-8  
 :  
 :  
 U 1  
  
 U 1  
 U 1  
 U 1  
 U 1  
 U 1

```

:26580 .PAGE "Test 30 MISMATCH LOGIC TEST (M8388 IDC MODULE)"
:26581 .++
:26582
:26583 .Test Description:
:26584
:26585 .This test is designed to insure that the mismatch logic of the DSK
:26586 .RD SEQ REG PAL is functioning correctly. A drive with an RL02 or no
:26587 .disk will be selected. Because of the design of the diskless loop,
:26588 .DAR bit <9> is compared to RD DATA bit <0>.
:26589 .The data pattern will be loaded into FIFO and the DAR. The data will
:26590 .then be taken from the FIFO and shifted through the diskless loop as
:26591 .read data. The data will be serialized through the DAR SERIALIZER PALS.
:26592 .The DAR data and the read data will then be compared in the DSK RD
:26593 .SEQ REG PAL. The IDC will then do a test match. BEN2/MISMATCH L will
:26594 .be selected. If MISMATCH L is asserted then the DAR will be incremented
:26595 .once. If MISMATCH L is not asserted (a MATCH was detected), then the
:26596 .DAR will be incremented twice. The CPU will then check the contents
:26597 .of the DAR.
:26598
:26599 .Data Patterns (2 bits):
:26600
:26601 .
:26602 . DAR READ DATA
:26603 .Pattern 1: 000 01
:26604 .Pattern 2: 200 00
:26605
:26606 .Assumptions:
:26607
:26608 .It is assumed that the diskless loop is fully operational and that
:26609 .the HEADER MATCH TEST and the CLEAR MATCH TEST have been run
:26610 .successfully.
:26611 .Test Steps:
:26612
:26613 .1. Set up error mask, error number and module number in LS
:26614 . (for error printouts) and clear the previous error number in LS.
:26615 .2. Force the microcode to DIAG.IDLE.
:26616 .3. If the drive sizing routine has not been run previously, JSR to
:26617 . DRIVE.SIZE.
:26618 .4. Set up the CSR to branch to the correct IDC microcode routine.
:26619
:26620
:26621 . *****
:26622 . * IDC *
:26623 . *****
:26624
:26625 . IDC1. Clear match.
:26626 . IDC2. Send XFER REQ to flag the CPU.
:26627
:26628 .5. Wait for a XFER.REQ from IDC.
:26629 .6. Clear CSR <2:0> and issue XFER GRANT to clear the XFER.REQ.
:26630 .7. Select an RL02 drive.
:26631 .8. Load the data pattern into the DAR.
:26632 .9. Load the data pattern into the FIFO.
:26633 .10. Set up the CSR to branch to the correct IDC microcode routine.
:26634

```

```

:26635 :
:26636 :
:26637 :
:26638 :
:26639 :
:26640 :
:26641 :
:26642 :
:26643 :
:26644 :
:26645 :
:26646 :
:26647 :
:26648 :
:26649 :
:26650 :
:26651 :
:26652 :
:26653 :
:26654 :
:26655 :
:26656 :
:26657 :
:26658 :
:26659 :
:26660 :
:26661 :
:26662 :
:26663 :
:26664 :
:26665 :
:26666 :
:26667 :
:26668 :
:26669 :
:26670 :
:26671 :
:26672 :
:26673 :
:26674 :
:26675 :
:26676 :
:26677 :
:26678 :
:26679 :
:26680 :
:26681 :
:26682 :
:26683 :
:26684 :
:26685 :
:26686 :
:26687 :
:26688 :
:26689 :

```

\*\*\*\*\*  
\* IDC \*  
\*\*\*\*\*

IDC3. Select the disk clock.  
 IDC4. Shift out the sync character.  
 IDC5. Follow with the data from the FIFO.  
 IDC6. Assert UENB BC H so that the DAR will begin serializing after sync is seen.  
 IDC7. Execute an instruction with UCON <1:0> = 10, TEST MATCH.  
 IDC8. Select the CPU clock.  
 IDC9. Select BEN2/MISMATCH L.  
 IDC10. If MISMATCH L is asserted then increment the DAR once. If MISMATCH L is not asserted (a MATCH was detected) increment the DAR twice.  
 IDC11. Issue XFER.REQ to flag the CPU.

11. Wait for a XFER.REQ from IDC.
12. Clear CSR <2:0> and issue XFER GRANT to clear the XFER.REQ.
13. Check the DAR = 1.
14. Increment the error number.
15. Repeat Steps 4 - 12 for Data Pattern 2.
16. Check the DAR = 201. (The DAR was loaded with 200 and should be incremented once)
17. Clear Match.
18. Clear IDC.
19. End test.

Errors:

| Printout:                                                               | EXPECTED<br>DAR | RECEIVED<br>DAR |
|-------------------------------------------------------------------------|-----------------|-----------------|
| ERROR 1 - The mismatch logic failed with DAR = 000 and READ DATA = 001. |                 |                 |
| ERROR 2 - The mismatch logic failed with DAR = 200 and READ DATA = 000. |                 |                 |

Debug:

ERROR 1 -

The data pattern is loaded into the DAR and into FIFO A address 0. A line with no disk drive attached or with an RL02 drive attached is selected and will assert RL02 H and deassert R80 H. The IDC will branch to a routine that will load the data from FIFO A address 0, into the DSR. This data is then shifted through the RL02 diskless loop to become READ DATA H (see RL02 DATA PATH TESTS if diskless loop is not functioning correctly). READ DATA H is an input to the SERIAL DAR LO PAL and it becomes M.READ DATA H. This signal is an input to the DSK RD SEQ REG PAL. As this data is being shifted through the diskless loop, the DAR data is being serialized. The BIT COUNTER PAL is cleared when ENB CLR BC H



[illegible]

```

:26713
:26714
:26715
U 16C5, B65E,15 :26716
:26717
U 16C6, 3E80,15 :26718
:26719
:26720
:26721
U 16C7, 10E0,15 :26722
:26723
U 16C8, 896C,84 :26724
:26725
:26726
U 16C9, 65A0,15 :26727
:26728
:26729
U 16CA, B64A,15 :26730
U 16CB, 3E8C,15 :26731
:26732
:26733
U 16CC, B640,15 :26734
U 16CD, BE82,15 :26735
:26736
U 16CE, 3776,15 :26737
U 16CF, 3E8A,15 :26738
:26739
U 16D0, B78E,15 :26740
:26741
U 16D1, D94E,35 :26742
U 16D2, 096D,41 :26743
:26744
:--
:*****
T.30:
 MOV LS[BEGIN.TEST] TO WR[0]
 MOV WR[0] TO LS[CONTROL.STATUS]
 MISC [SET.CP.ATTN]
WAIT.T30.0:
 JMP [WAIT.T30.0]
 CLR LS[PREVIOUS.ERROR]
 MOV LS[IDC] TO WR[0]
 MOV WR[0] TO LS[MODULE.NUM]
 MOV LS[#1] TO WR[0]
 MOV WR[0] TO LS[ERROR.NUMBER]
 MOV LS[DAR.MASK] TO WR[0]
 MOV WR[0] TO LS[ERROR.MASK]
 MOV LS[DRIVE.STATUS] TO WR[0]
 BIT LS[BIT7] WITH WR[0],
 DT(LONG)&SET.ALU.CC
 JMP.IF[BIT.SET] TO [SIZE.OK.T30.1]
 ;SET BIT 15 IN WR[0] FOR
 ; CONTROL AND STATUS WORD
 ;SET BIT 15 IN CONTROL AND
 ; STATUS WORD - BIT 15
 ; INDICATES START OF TEST TO
 ; CONSOLE
 ;ISSUE CPU ATTN TO CONSOLE
 ;LOOP HERE WAITING FOR
 ; CONSOLE TO RESPOND
 ;CLEAR PREVIOUS ERROR NUMBER
 ; IN LS
 ;STORE MODULE CODE IN LS TO
 ; INDICATE IDC MODULE
 ;SET WRO = 1
 ;SET UP ERROR NUMBER = 1 IN LS
 ;GET THE MASK TO CHECK THE
 ; DAR
 ;HAS THE DRIVE SIZING ROUTINE
 ; BEEN RUN?
 ;IF YES THEN GO RUN TEST

```

```

ZZ-ENKCF-1.4 : ENKCF.MIC Test 30 MISMATCH L15-MAR-83 D 11 Fiche 3 Frame D11 Sequence 545
: ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40 Page 539
: ENKCF.MIC Test 30 MISMATCH LOGIC TEST (M8388 IDC MODULE)

U 16D3, 8870,0C :26745 JSR [DRIVE.SIZE] ;GO RUN THE SIZING ROUTINE
:26746
:26747 SIZE.OK.T30.1:
U 16D4, 8872,EC :26748 JSR [DIAG.CODE] ;FORCE THE IDC TO DIAG.IDLE
:26749
:26750 ;SELECT AN RL02 DRIVE AND SAVE THE INFO IN WR[2]
:26751
U 16D5, 878E,15 :26752 MOV LS[DRIVE.STATUS] TO WR[0]
:26753 BIT LS[BIT0] WITH WR[0], ;IS DRIVE 0 AN R80?
:26754 DT(LONG)&SET.ALU.CC
U 16D6, 5940,35 :26755 JMP.IF[BIT0] TO [SET.DS0.T30.1] ;JMP IF DRV 0 IS NOT AN R80
U 16D7, 096D,A9 :26756 MOV LS[SETCSR.DS1] TO WR[2] ;SAVE THE DATA TO SEL DRIVE 1
U 16D8, 8717,15 :26757 JMP [TEST.T30.1]
U 16D9, 096D,B4 :26758 SET.DS0.T30.1:
:26759 MOV LS[SETCSR.DS0] TO WR[2] ;SAVE THE DATA TO SEL DRIVE 0
:26760
:26761 TEST.T30.1:
:26762 LOOP.T30.1:
U 16DB, 370E,15 :26763 MOV LS[SETCSR.F7] TO WR[0]
U 16DC, 17A0,15 :26764 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:26765 ;MAINT=0,CRDY=0,F<2:0>=111
U 16DD, 379E,15 :26766 MOV LS[SETCSR.R80] TO WR[0] ;SET UP THE CSR TO CONTINUE TO
U 16DE, 3700,95 :26767 MOV LS[SETCSR.F0] TO WR[1] ; BRANCH TO THE CORRECT ROUTINE
U 16DF, AEC2,15 :26768 BIS WR[1] TO WR[0]
U 16E0, 17A0,15 :26769 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:26770 ;F<2:0> = 000
:26771 ;UCLR MATCH
:26772 -----
:26773 -----
:26774 *****
:26775 * IDC *
:26776 *****
:26777 -----
:26778 018:
:26779 =000
:26780 DIAG.IDLE:
:26781
:26782 BEN0/F0,
:26783 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:26784 BEN2/F2,
:26785 NAD/DIAG.IDLE
:26786 -----
:26787 01F:
:26788 =111
:26789
:26790 BEN0/CRDY.L,
:26791 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:26792 NAD/TEST.FUNC7 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:26793 -----
:26794 079:
:26795 =0*1
:26796
:26797 NAD/GROUP1
:26798 -----
:26799 182:

```

ZZ-ENKCF-1.4  
: ENKCF.MCR  
: ENKCF.MIC

ENKCF.MIC

MICRO2 1M(01)

Test 30 MISMATCH LOGIC TEST (M8388 IDC MODULE)

E 11

15-MAR-83 11:46:22

Fiche 3 Frame E11

Sequence 546

Rev 01.40

Page 540

ZZ-f

:

```
:26800 : =0**
:26801 : GROUP1:
:26802 :
:26803 : BEN2/R80.FORMAT.H, ; WHEN THE R80 FORMAT BIT SETS..BRANCH
:26804 : NAD/GROUP1 ; TO NEXT WORD AND THEN BRANCH ON THE
:26805 : ; FUNCTION BITS
:26806 : -----
:26807 : 186:
:26808 : =1**
:26809 :
:26810 : BEN0/F0,
:26811 : BEN1/F1, ; BRANCH ON THE CSR F<2:0> BITS
:26812 : BEN2/F2,
:26813 : NAD/GROUP1.F0
:26814 : -----
:26815 : 028:
:26816 : =000
:26817 : GROUP1.F0:
:26818 :
:26819 : CLR.MATCH.1:
:26820 :
:26821 : UCON/CLR.MATCH, ; CLR THE MATCH CONDITION
:26822 : NAD/TEST.MATCH.1
:26823 : -----
:26824 : 1F6:
:26825 : TEST.MATCH.1:
:26826 :
:26827 : BEN2/MISMATCH.L, ; IS MATCH ASSERTED?
:26828 : NAD/BEN2.1
:26829 : -----
:26830 : 1A0:
:26831 : =0**
:26832 : BEN2.1:
:26833 :
:26834 : INCR.DAR/ON, ; INCR DAR
:26835 : NAD/XFER.REQ ; FLAG THE CPU
:26836 : -----
:26837 : 1A4:
:26838 : =1**
:26839 : BEN2.2:
:26840 :
:26841 : INCR.DAR/ON, ; INCR DAR IF BEN2 CONDITION ASSERTED
:26842 : NAD/BEN2.1 ; INCR DAR AGAIN IF BEN2 COND ASSERTED
:26843 : -----
:26844 : 1D8:
:26845 : XFER.REQ:
:26846 : UCMD/SET.XFER.RQST, ; SEND XFER REQ TO THE CPU
:26847 : NAD/DIAG.WAIT
:26848 : -----
:26849 : 1D7:
:26850 : DIAG.WAIT:
:26851 :
:26852 : BEN1/XFER.REQ, ; WAIT FOR THE CPU TO ISSUE XFER GRANT
:26853 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:26854 : -----
```

```

:26855 :159:
:26856 :=*0*
:26857 :DIAG.WAIT1:
:26858 :
:26859 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:26860 : ; COMMAND FROM THE CPU
:26861 :-----
:26862 :15B:
:26863 :=*1*
:26864 :DIAG.WAIT2:
:26865 :
:26866 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:26867 : ; WAIT SOME MORE
:26868 :-----
:26869 :WAIT.IDC.T30.1:
:26870 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ FROM IDC
:26871 : JMP [IDC.DONE.T30.1] ;XFER REQ SET
:26872 : JMP [WAIT.IDC.T30.1] ;WAIT SOME MORE
:26873 :
:26874 :IDC.DONE.T30.1:
:26875 : MOV LS[SETCSR.F0] TO WR[0] ;CLEAR CSR F<2:0>
:26876 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:26877 : MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
:26878 :
:26879 : CLR WR[0] ;GET THE DATA TO WRITE TO DAR
:26880 : PORT.WRITE PORT.ADRS[DAR] WITH WR[0] ;CLR THE DAR
:26881 :
:26882 : MOV LS[#1] TO WR[0] ;GET THE DATA TO WRITE TO FIFOA
:26883 : PORT.CONTROL [SEL.FIFO.A] ;SELECT FIFO A
:26884 : PORT.CONTROL [CLEAR.FIFO.CNTR] ;SET FIFO A ADRS=0
:26885 : PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;WRITE THE DATA TO FIFO A
:26886 :
:26887 : PORT.CONTROL [CLEAR.FIFO.CNTR] ;SET FIFO A ADRS=0
:26888 :
:26889 :;SET UP THE CSR TO BRANCH TO THE CORRECT IDC ROUTINE
:26890 :
:26891 : MOV LS[SETCSR.F7] TO WR[0]
:26892 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:26893 :
:26894 : MOV LS[SETCSR.R80] TO WR[0] ;MAINT=0,CRDY=0,F<2:0>=111
:26895 : MOV LS[SETCSR.F2] TO WR[1] ;SET UP THE CSR TO CONTINUE TO
:26896 : BIS WR[1] TO WR[0] ; BRANCH TO THE CORRECT ROUTINE
:26897 : BIS WR[2] TO WR[0]
:26898 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;INCLUDE THE RL02 DRV SELECT
:26899 : ;WRITE THE CSR
:26900 :-----
:26901 :-----
:26902 : *****
:26903 : * IDC *
:26904 : *****
:26905 :-----
:26906 :018:
:26907 :=000
:26908 :DIAG.IDLE:
:26909 :

```

U 16E1, DB00,1A  
 U 16E2, 096E,44  
 U 16E3, 096E,14

U 16E4, B700,15  
 U 16E5, 17A0,15  
 U 16E6, 90FA,15

U 16E7, 2F80,15  
 U 16E8, 97A4,15

U 16E9, B640,15  
 U 16EA, 17D8,15  
 U 16EB, 17C0,15  
 U 16EC, 97A8,15

U 16ED, 17C0,15

U 16EE, 370E,15  
 U 16EF, 17A0,15

U 16F0, 379E,15  
 U 16F1, B704,95  
 U 16F2, AEC2,15  
 U 16F3, AEC4,15  
 U 16F4, 17A0,15

ZZ-1  
 ;  
 ;

U 1  
 U 1  
 U 1

U 1  
 U 1  
 U 1

U 1

ZZ-ENKCF-1.4  
: ENKCF.MCR  
: ENKCF.MIC

ENKCF.MIC

Test 30 MISMATCH LOGIC TEST (M8388 IDC MODULE)  
MICRO2 1M(01) 15-MAR-83 11:46:22  
Test 30 MISMATCH LOGIC TEST (M8388 IDC MODULE)

G 11  
Fiche 3 Frame G11  
ENKCF Micro-diagnostic for IDC module

Sequence 548  
Rev 01.40 Page 542

ZZ-E  
:  
:

```
:26910 : BEN0/F0,
:26911 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:26912 : BEN2/F2,
:26913 : NAD/DIAG.IDLE
:26914 :-----
:26915 : 01F:
:26916 : =111
:26917 :
:26918 : BEN0/CRDY.L,
:26919 : BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:26920 : NAD/TEST.FUNC7 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:26921 :-----
:26922 : 079:
:26923 : =0*1
:26924 :
:26925 : NAD/GROUP1
:26926 :-----
:26927 : 182:
:26928 : =0**
:26929 : GROUP1:
:26930 :
:26931 : BEN2/R80.FORMAT.H, ;WHEN THE R80 FORMAT BIT SETS..BRANCH
:26932 : NAD/GROUP1 ; TO NEXT WORD AND THEN BRANCH ON THE
:26933 : ; FUNCTION BITS
:26934 :-----
:26935 : 186:
:26936 : =1**
:26937 :
:26938 : BEN0/F0,
:26939 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:26940 : BEN2/F2,
:26941 : NAD/GROUP1.F0
:26942 :-----
:26943 : 02A:
:26944 : =010
:26945 : MATCH.OR.WRTCHK.1:
:26946 :
:26947 : UMAINT/ON,
:26948 : LOOP.LOCK/ON, ;MAINT PREVENTS LOOP LOCK FROM SELECTING
:26949 : DISK.WRITE/ON, ;READ DATA...DISK WRITE STILL SELECTS DSRO
:26950 : LOAD.LOOP.CNTR.[0E0],
:26951 : NAD/MATCH.OR.WRTCHK.2
:26952 :-----
:26953 : 198:
:26954 : =0
:26955 : MATCH.OR.WRTCHK.2:
:26956 :
:26957 : UMAINT/ON,
:26958 : LOOP.LOCK/ON, ;SHIFT OUT 32 ZEROS TO SYNC UP
:26959 : DISK.WRITE/ON, ; DATA SEPARATOR
:26960 : UCMD/SET.FIFOA,
:26961 : INCR.LPCNTR/ON,
:26962 : BEN0/CNTR.OVFLW,
:26963 : NAD/MATCH.OR.WRTCHK.2
:26964 :-----
```

```

:26965 :199:
:26966 :=1
:26967 :MATCH.OR.WRTCHK.3:
:26968 :
:26969 : UMAINT/ON,
:26970 : LOOP.LOCK/ON,
:26971 : DISK.WRITE/ON, ;MUST WAIT 3 CYCLES FOR SYNC SEEN
:26972 : ENB.DSK.CLK,
:26973 : ENB.CLR.BC/ON, ;CLR THE BIT COUNTER WHEN SYNC SEEN
:26974 : NAD/MATCH.OR.WRTCHK.4 ; FOR THE MATCH TEST
:26975 :-----
:26976 :1F2:
:26977 :MATCH.OR.WRTCHK.4:
:26978 :
:26979 : UMAINT/ON,
:26980 : LOOP.LOCK/ON, ;WAIT SOME MORE
:26981 : DISK.WRITE/ON,
:26982 : ENB.CLR.BC/ON, ;CLR THE BIT COUNTER WHEN SYNC SEEN
:26983 : NAD/MATCH.OR.WRTCHK.5 ; FOR THE MATCH TEST
:26984 :-----
:26985 :1F3:
:26986 :MATCH.OR.WRTCHK.5:
:26987 :
:26988 : UMAINT/ON,
:26989 : LOOP.LOCK/ON, ;WAIT SOME MORE
:26990 : DISK.WRITE/ON,
:26991 : ENB.CLR.BC/ON, ;CLR THE BIT COUNTER WHEN SYNC SEEN
:26992 : NAD/MATCH.OR.WRTCHK.6 ; FOR THE MATCH TEST
:26993 :-----
:26994 :1F4:
:26995 :MATCH.OR.WRTCHK.6:
:26996 :
:26997 : UMAINT/ON,
:26998 : LOOP.LOCK/ON,
:26999 : DISK.WRITE/ON, ;LOAD THE DATA FROM FIFO A INTO THE DSR
:27000 : ENB.FIFO/ON,
:27001 : LOAD.DSR/ON,
:27002 : BEN2/MAINT, ;MATCH TEST OR WRTCHK TEST?
:27003 : NAD/MATCH.OR.WRTCHK.7
:27004 :-----
:27005 :183:
:27006 :=0**
:27007 :MATCH.OR.WRTCHK.7:
:27008 :
:27009 : UMAINT/ON,
:27010 : LOOP.LOCK/ON, ;MATCH TEST
:27011 : DISK.WRITE/ON,
:27012 : LOAD.LOOP.CNTR.[OF6],
:27013 : NAD/MATCH.1
:27014 :-----
:27015 :19A:
:27016 :=0
:27017 :MATCH.1:
:27018 :
:27019 : UMAINT/ON,

```

[illegible]

ZZ-ENKCF-1.4  
: ENKCF.MCR  
: ENKCF.MIC

ENKCF.MIC

MICRO2 1M(01)

Test 30 MISMATCH L15-MAR-83

15-MAR-83 11:46:22

Test 30 MISMATCH LOGIC TEST (M8388 IDC MODULE)

J 11

Fiche 3 Frame J11

Sequence 551

Rev 01.40

Page 545

ZZ-E  
:  
:

```
:27075 :-----
:27076 :159:
:27077 :=*0*
:27078 :DIAG.WAIT1:
:27079
:27080 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:27081 : ; COMMAND FROM THE CPU
:27082 :-----
:27083 :15B:
:27084 :=*1*
:27085 :DIAG.WAIT2:
:27086
:27087 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:27088 : ; WAIT SOME MORE
:27089 :-----
:27090 :WAIT.IDC.T30.2:
:27091 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ FROM IDC
:27092 : JMP [IDC.DONE.T30.2] ;XFER REQ SET
:27093 : JMP [WAIT.IDC.T30.2] ;WAIT SOME MORE
:27094
:27095 :IDC.DONE.T30.2:
:27096 : MOV LS[SETCSR.F0] TO WR[0] ;CLEAR CSR F<2:0>
:27097 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:27098 : MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
:27099
:27100 :READ.DAR.T30.1:
:27101 : MISC [SET.PORT.I.0] ;SELECT THE IDC
:27102 : PORT.READ PORT.ADRS[DAR] ;SET UP TO READ THE DAR
:27103 : MOV PORT TO LS[PORT0] ;READ THE DATA FROM DAR AND
:27104 : ; SAVE IT IN LS
:27105 : MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
:27106 : MOV LS[#1] TO WR[1] ;SET UP EXPECTED DATA
:27107 : JSR [CHECK.RESULT] ;CHECK THE DATA
:27108 : JMP [LOOP.T30.1] ;LOOP HERE IF ERR & LOE IS SET
:27109
:27110 : INC LS[ERROR.NUMBER] ;ERROR 2
:27111
:27112 : ;SET UP THE CSR TO FORCE THE IDC TO CLR MATCH
:27113
:27114 :LOOP.T30.2:
:27115 : MOV LS[SETCSR.F7] TO WR[0] ;WRITE THE CSR
:27116 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;MAINT=0,CRDY=0,F<2:0>=111
:27117 : ;SET UP THE CSR TO CONTINUE TO
:27118 : MOV LS[SETCSR.R80] TO WR[0] ;BRANCH TO THE CORRECT ROUTINE
:27119 : MOV LS[SETCSR.F0] TO WR[1]
:27120 : BIS WR[1] TO WR[0]
:27121 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:27122 : ;F<2:0> = 000
:27123 :UCLR MATCH
:27124 :-----
:27125 :-----
:27126 : *****
:27127 : * IDC *
:27128 : *****
:27129 :-----
```

U 17  
U 17

U 17  
U 17  
U 17  
U 17



```

:27130 :018:
:27131 :=000
:27132 :DIAG.IDLE:
:27133 :
:27134 : BEN0/F0,
:27135 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:27136 : BEN2/F2,
:27137 : NAD/DIAG.IDLE
:27138 :-----
:27139 :01F:
:27140 :=111
:27141 :
:27142 : BEN0/CRDY.L,
:27143 : BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:27144 : NAD/TEST.FUNC7 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:27145 :-----
:27146 :079:
:27147 :=0*1
:27148 :
:27149 : NAD/GROUP1
:27150 :-----
:27151 :182:
:27152 :=0**
:27153 :GROUP1:
:27154 :
:27155 : BEN2/R80.FORMAT.H, ;WHEN THE R80 FORMAT BIT SETS..BRANCH
:27156 : NAD/GROUP1 ; TO NEXT WORD AND THEN BRANCH ON THE
:27157 : ; FUNCTION BITS
:27158 :-----
:27159 :186:
:27160 :=1**
:27161 :
:27162 : BEN0/F0,
:27163 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:27164 : BEN2/F2,
:27165 : NAD/GROUP1.F0
:27166 :-----
:27167 :028:
:27168 :=000
:27169 :GROUP1.F0:
:27170 :
:27171 :CLR.MATCH.1:
:27172 :
:27173 : UCON/CLR.MATCH, ;CLR THE MATCH CONDITION
:27174 : NAD/TEST.MATCH.1
:27175 :-----
:27176 :1F6:
:27177 :TEST.MATCH.1:
:27178 :
:27179 : BEN2/MISMATCH.L, ;IS MATCH ASSERTED?
:27180 : NAD/BEN2.1
:27181 :-----
:27182 :1A0:
:27183 :=0**
:27184 :BEN2.1:

```

U 1711, 2F80, 15  
U 1712, 17D8, 15  
U 1713, 17C0, 15  
U 1714, 97A8, 15

```

U 1715, 17C0,15 :27240 PORT.CONTROL [CLEAR.FIFO.CNTR] ;SET FIFO A ADRS=0
 :27241
 :27242 ;SET UP THE CSR TO BRANCH TO THE CORRECT IDC ROUTINE
 :27243
U 1716, 370E,15 :27244 MOV LS[SETCSR.F7] TO WR[0]
U 1717, 17A0,15 :27245 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
 :27246 ;MAINT=0,CRDY=0,F<2:0>=111
 :27247 MOV LS[SETCSR.R80] TO WR[0] ;SET UP THE CSR TO CONTINUE TO
U 1718, 379E,15 :27248 MOV LS[SETCSR.F2] TO WR[1] ; BRANCH TO THE CORRECT ROUTINE
U 1719, B704,95 :27249 BIS WR[1] TO WR[0]
U 171A, AEC2,15 :27250 BIS WR[2] TO WR[0] ;INCLUDE THE RLO2 DRV SELECT
U 171B, AEC4,15 :27251 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
U 171C, 17A0,15 :27252
 :27253
 :27254
 :27255 *****
 :27256 * IDC *
 :27257 *****
 :27258
 :27259 018:
 :27260 =000
 :27261 DIAG.IDLE:
 :27262
 :27263 BEN0/F0,
 :27264 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
 :27265 BEN2/F2,
 :27266 NAD/DIAG.IDLE
 :27267
 :27268 01F:
 :27269 =111
 :27270
 :27271 BEN0/CRDY.L,
 :27272 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
 :27273 NAD/TEST.FUNC? ; DEPENDENT ON CSR <CRDY> AND <MAINT>
 :27274
 :27275 079:
 :27276 =0*1
 :27277
 :27278 NAD/GROUP1
 :27279
 :27280 182:
 :27281 =0**
 :27282 GROUP1:
 :27283
 :27284 BEN2/R80.FORMAT.H, ;WHEN THE R80 FORMAT BIT SETS..BRANCH
 :27285 NAD/GROUP1 ; TO NEXT WORD AND THEN BRANCH ON THE
 :27286 ; FUNCTION BITS
 :27287
 :27288 186:
 :27289 =1**
 :27290
 :27291 BEN0/F0,
 :27292 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
 :27293 BEN2/F2,
 :27294 NAD/GROUP1.F0

```

```
:27295 :-----
:27296 :02A:
:27297 :=010
:27298 :MATCH.OR.WRTCHK.1:
:27299 :
:27300 : UMAINT/ON,
:27301 : LOOP.LOCK/ON, ;MAINT PREVENTS LOOP LOCK FROM SELECTING
:27302 : DISK.WRITE/ON, ;READ DATA...DISK WRITE STILL SELECTS DSRO
:27303 : LOAD.LOOP.CNTR.[0E0],
:27304 : NAD/MATCH.OR.WRTCHK.2
:27305 :-----
:27306 :198:
:27307 :=0
:27308 :MATCH.OR.WRTCHK.2:
:27309 :
:27310 : UMAINT/ON,
:27311 : LOOP.LOCK/ON, ;SHIFT OUT 32 ZEROS TO SYNC UP
:27312 : DISK.WRITE/ON, ; DATA SEPARATOR
:27313 : UCMD/SET.FIFOA,
:27314 : INCR.LPCNTR/ON,
:27315 : BENO/CNTR.OVFLW,
:27316 : NAD/MATCH.OR.WRTCHK.2
:27317 :-----
:27318 :199:
:27319 :=1
:27320 :MATCH.OR.WRTCHK.3:
:27321 :
:27322 : UMAINT/ON,
:27323 : LOOP.LOCK/ON,
:27324 : DISK.WRITE/ON, ;MUST WAIT 3 CYCLES FOR SYNC SEEN
:27325 : ENB.DSK.CLK,
:27326 : ENB.CLR.BC/ON, ;CLR THE BIT COUNTER WHEN SYNC SEEN
:27327 : NAD/MATCH.OR.WRTCHK.4 ; FOR THE MATCH TEST
:27328 :-----
:27329 :1F2:
:27330 :MATCH.OR.WRTCHK.4:
:27331 :
:27332 : UMAINT/ON,
:27333 : LOOP.LOCK/ON, ;WAIT SOME MORE
:27334 : DISK.WRITE/ON,
:27335 : ENB.CLR.BC/ON, ;CLR THE BIT COUNTER WHEN SYNC SEEN
:27336 : NAD/MATCH.OR.WRTCHK.5 ; FOR THE MATCH TEST
:27337 :-----
:27338 :1F3:
:27339 :MATCH.OR.WRTCHK.5:
:27340 :
:27341 : UMAINT/ON,
:27342 : LOOP.LOCK/ON, ;WAIT SOME MORE
:27343 : DISK.WRITE/ON,
:27344 : ENB.CLR.BC/ON, ;CLR THE BIT COUNTER WHEN SYNC SEEN
:27345 : NAD/MATCH.OR.WRTCHK.6 ; FOR THE MATCH TEST
:27346 :-----
:27347 :1F4:
:27348 :MATCH.OR.WRTCHK.6:
:27349 :
```

```

:27350 : UMAINT/ON,
:27351 : LOOP.LOCK/ON,
:27352 : DISK.WRITE/ON, ;LOAD THE DATA FROM FIFO A INTO THE DSR
:27353 : ENB.FIFO/ON,
:27354 : LOAD.DSR/ON,
:27355 : BEN2/MAINT, ;MATCH TEST OR WRTCHK TEST?
:27356 : NAD/MATCH.OR.WRTCHK.7
:27357 :-----
:27358 :183:
:27359 :=0**
:27360 :MATCH.OR.WRTCHK.7:
:27361 :
:27362 : UMAINT/ON,
:27363 : LOOP.LOCK/ON, ;MATCH TEST
:27364 : DISK.WRITE/ON,
:27365 : LOAD.LOOP.CNTR.[OF6],
:27366 : NAD/MATCH.1
:27367 :-----
:27368 :19A:
:27369 :=0
:27370 :MATCH.1:
:27371 :
:27372 : UMAINT/ON,
:27373 : LOOP.LOCK/ON,
:27374 : DISK.WRITE/ON, ;WAIT FOR DATA TO SHIFT THRU DISKLESS
:27375 : INCR.LPCNTR/ON, ; LOOP TO BECOME RD DATA
:27376 : BEN0/CNTR.OVFLW,
:27377 : NAD/MATCH.1
:27378 :-----
:27379 :19B:
:27380 :=1
:27381 :MATCH.2:
:27382 :
:27383 : UMAINT/ON, ;KEEP THE CLK FROM THE DATA SEPARATOR
:27384 : LOOP.LOCK/ON, ; GOING AND TEST MATCH
:27385 : DISK.WRITE/ON,
:27386 : UCON/TEST.MATCH,
:27387 : NAD/MATCH.3
:27388 :-----
:27389 :1F5:
:27390 :MATCH.3:
:27391 :
:27392 : UMAINT/ON, ;KEEP THE CLK FROM THE DATA SEPARATOR
:27393 : LOOP.LOCK/ON, ; GOING
:27394 : DISK.WRITE/ON,
:27395 : ENB.CPU.CLK, ;SELECT THE CPU CLK
:27396 : NAD/TEST.MATCH.1
:27397 :-----
:27398 :1F6:
:27399 :TEST.MATCH.1:
:27400 :
:27401 : BEN2/MISMATCH.L, ;IS MATCH ASSERTED?
:27402 : NAD/BEN2.1
:27403 :-----
:27404 :1A0:

```

```

:27405 :=0**
:27406 :BEN2.1:
:27407
:27408 INCR.DAR/ON, ;INCR DAR
:27409 NAD/XFER.REQ ;FLAG THE CPU
:27410 -----
:27411 :1A4:
:27412 :=1**
:27413 :BEN2.2:
:27414
:27415 INCR.DAR/ON, ;INCR DAR IF BEN2 CONDITION ASSERTED
:27416 NAD/BEN2.1 ;INCR DAR AGAIN IF BEN2 COND ASSERTED
:27417 -----
:27418 :1D8:
:27419 :XFER.REQ:
:27420 UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:27421 NAD/DIAG.WAIT
:27422 -----
:27423 :1D7:
:27424 :DIAG.WAIT:
:27425
:27426 BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:27427 NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:27428 -----
:27429 :159:
:27430 :=*0*
:27431 :DIAG.WAIT1:
:27432
:27433 NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:27434 ; COMMAND FROM THE CPU
:27435 -----
:27436 :15B:
:27437 :=*1*
:27438 :DIAG.WAIT2:
:27439
:27440 NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:27441 ; WAIT SOME MORE
:27442 -----
:27443 :WAIT.IDC.T30.4:
:27444 SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ FROM IDC
:27445 JMP [IDC.DONE.T30.4] ;XFER REQ SET
:27446 JMP [WAIT.IDC.T30.4] ;WAIT SOME MORE
:27447
:27448 :IDC.DONE.T30.4:
:27449 MOV LS[SETCSR.F0] TO WR[0] ;CLEAR CSR F<2:0>
:27450 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:27451 MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
:27452
:27453 :READ.DAR.T30.2:
:27454 MISC [SET.PORT.I.0] ;SELECT THE IDC
:27455 PORT.READ PORT.ADRS[DAR] ;SET UP TO READ THE DAR
:27456 MOV PORT TO LS[PORT0] ;READ THE DATA FROM DAR AND
:27457 ; SAVE IT IN LS
:27458 MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
:27459 MOV LS[BIT9] TO WR[1] ;SET UP EXPECTED DATA

```

U 171D, DB00,1A  
 U 171E, 0972,04  
 U 171F, 8971,D4

U 1720, B700,15  
 U 1721, 17A0,15  
 U 1722, 90FA,15

U 1723, 9090,15  
 U 1724, 1784,15  
 U 1725, 3B32,15

U 1726, 3732,15  
 U 1727, 3652,95

ZZ-E  
 ;  
 ;

U 17  
 U 17  
 U 17

U 17  
 U 17  
 U 17

U 17  
 U 17  
 U 17

U 17  
 U 17  
 U 17  
 U 17  
 U 17

U 17  
 U 17

U 17

U 17  
 U 17  
 U 17

U 17

U 17  
 U 17

U 17  
 U 17  
 U 17  
 U 17

```

U 1728, 4740,95 :27460 BIS LS[BIT0] TO WR[1]
U 1729, 0869,3C :27461 JSR [CHECK.RESULT] ;CHECK THE DATA
U 172A, 8970,34 :27462 JMP [LOOP.T30.2] ;LOOP HERE IF ERR & LOE IS SET
:27463
:27464 CLEANUP.T30:
:27465 ;CLR MATCH AS PART OF CLEANUP
U 172B, 370E,15 :27466 MOV LS[SETCSR.F7] TO WR[0]
U 172C, 17A0,15 :27467 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:27468 ;MAINT=0,CRDY=0,F<2:0>=111
U 172D, 379E,15 :27469 MOV LS[SETCSR.R80] TO WR[0] ;SET UP THE CSR TO CONTINUE TO
U 172E, 3700,95 :27470 MOV LS[SETCSR.F0] TO WR[1] ; BRANCH TO THE CORRECT ROUTINE
U 172F, AEC2,15 :27471 BIS WR[1] TO WR[0]
U 1730, 17A0,15 :27472 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:27473 ;F<2:0> = 000
:27474 UCLR MATCH
:27475 -----
:27476 -----
:27477 *****
:27478 * IDC *
:27479 *****
:27480 -----
:27481 018:
:27482 =000
:27483 DIAG.IDLE:
:27484
:27485 BEN0/F0,
:27486 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:27487 BEN2/F2,
:27488 NAD/DIAG.IDLE
:27489 -----
:27490 01F:
:27491 =111
:27492
:27493 BEN0/CRDY.L,
:27494 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:27495 NAD/TEST.FUNC7 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:27496 -----
:27497 079:
:27498 =0*1
:27499
:27500 NAD/GROUP1
:27501 -----
:27502 182:
:27503 =0**
:27504 GROUP1:
:27505
:27506 BEN2/R80.FORMAT.H, ;WHEN THE R80 FORMAT BIT SETS..BRANCH
:27507 NAD/GROUP1 ; TO NEXT WORD AND THEN BRANCH ON THE
:27508 ; FUNCTION BITS
:27509 -----
:27510 186:
:27511 =1**
:27512
:27513 BEN0/F0,
:27514 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS

```

|   |   |
|---|---|
| U | 1 |
| U | 1 |
| U | 1 |
|   |   |
| U | 1 |
| U | 1 |
| U | 1 |
|   |   |
| U | 1 |



```

:27570 ; ; WAIT SOME MORE
:27571 ; -----
:27572 WAIT.IDC.T30.5:
U 1731, DB00,1A :27573 SKIP.IF[NO.FAST.INTERUPT] ;WAIT FOR XFER REQ FROM IDC
U 1732, 0973,44 :27574 JMP [IDC.DONE.T30.5] ;XFER REQ SET
U 1733, 0973,14 :27575 JMP [WAIT.IDC.T30.5] ;WAIT SOME MORE
:27576
:27577 IDC.DONE.T30.5:
U 1734, B700,15 :27578 MOV LS[SETCSR.F0] TO WR[0] ;CLEAR CSR F<2:0>
U 1735, 17A0,15 :27579 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
U 1736, 90FA,15 :27580 MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
:27581
:27582
U 1737, 8875,3C :27583 JSR [CLEANUP] ;CLEANUP CONDITIONS SET BY THIS
:27584 ; TEST
:27585 END.T30: ;END TEST
:27586
:27587
:27588
:27589
:27590

```

:27591 .PAGE 'Test 31 DAR SERIALIZER AND RLO2 HEADER MATCH TEST (M8388 IDC MODULE)'  
:27592 :++

:27593  
:27594  
:27595

:27596 : Test Description:

:27597

:27598

:27599

:27600

:27601

:27602

:27603

:27604

:27605

:27606

:27607

:27608

:27609

:27610

:27611

:27612

:27613

:27614

:27615

:27616

:27617

:27618

:27619

:27620 : Assumptions:

:27621

:27622

:27623

:27624

:27625

:27626

:27627

:27628

:27629

:27630

:27631

:27632

:27633

:27634

:27635

:27636

:27637

:27638

:27639

:27640

:27641

:27642

:27643

:27644

:27645

This test is designed to insure that the DAR serializer functions correctly when an RLO2 drive is selected for bits <15:9> of the DAR. This will be accomplished by loading the data pattern into the DAR and into the FIFO. The data will be shifted out of the FIFO and through the diskless loop to become READ DATA H. The DAR will be serialized the then compared bit by bit to the READ DATA H. Because of the design of the diskless loop, DAR bit <9> is compared with READ DATA <0>, DAR <10> is compared with READ DATA bit <1>, etc. When all bits have been compared, the IDC will execute a test match. If a match was detected, the DAR will be incremented twice. If the data did not match, the DAR will be incremented once. The CPU will then read and check the DAR.

Data Patterns:

Float a zero through a field of ones for bits <15:9> of the DAR.

Assumptions:

It is assumed that the diskless loop has been proven fully operational.

Test Steps:

1. Set up error mask, error number and module number in LS (for error printouts) and clear the previous error number in LS.
2. Force the IDC microcode to DIAG.IDLE.
3. If the drive sizing routine has not been run previously, JSR to DRIVE.SIZE.
4. Select an RLO2 drive.
5. Clear Match.
6. Load the data pattern into the DAR.
7. Load the data pattern into the FIFO.
8. Set up the CSR to branch to the correct IDC microcode routine.

\*\*\*\*\*  
\* IDC \*  
\*\*\*\*\*

- IDC1. Send the sync character.  
IDC2. Load the data pattern from the FIFO into the DSR.  
IDC3. Shift the data from the DSR and through the diskless loop.  
IDC4. Assert UENB CLR BC H so that the DAR will begin serializing after sync seen.  
IDC5. Compare the serialized DAR and READ DATA H.  
IDC6. Execute an instruction with UCON<1:0> = 10, test match.  
IDC7. Select the CPU clock.  
IDC8. Select BEN2/MISMATCH L.

[illegible]

U 17  
U 17  
U 17  
U 17  
U 17

U 17  
U 17

U 17  
U 17  
U 17  
U 17

```

:27756
:27757 TST.NXT.PAT.T31:
:27758 LOOP.T31.1:
U 1751, 370E,15 :27759 MOV LS[SETCSR.F7] TO WR[0]
U 1752, 17A0,15 :27760 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:27761 ;MAINT=0,CRDY=0,F<2:0>=111
:27762 ;CONTINUE TO BRANCH
U 1753, 379E,15 :27763 MOV LS[SETCSR.R80] TO WR[0] ;SET UP THE CSR TO CONTINUE TO
U 1754, 3700,95 :27764 MOV LS[SETCSR.F0] TO WR[1] ; BRANCH TO THE CORRECT ROUTINE
U 1755, AEC2,15 :27765 BIS WR[1] TO WR[0]
U 1756, 17A0,15 :27766 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:27767 ;F<2:0> = 000
:27768 ;UCLR MATCH
:27769 -----
:27770 -----
:27771 : *****
:27772 : * IDC *
:27773 : *****
:27774 -----
:27775 :018:
:27776 :=000
:27777 :DIAG.IDLE:
:27778 :
:27779 : BEN0/F0,
:27780 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:27781 : BEN2/F2,
:27782 : NAD/DIAG.IDLE
:27783 -----
:27784 :01F:
:27785 :=111
:27786 :
:27787 : BEN0/CRDY.L,
:27788 : BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:27789 : NAD/TEST.FUNC7 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:27790 -----
:27791 :079:
:27792 :=0*1
:27793 :
:27794 : NAD/GROUP1
:27795 -----
:27796 :182:
:27797 :=0**
:27798 :GROUP1:
:27799 :
:27800 : BEN2/R80.FORMAT.H, ;WHEN THE R80 FORMAT BIT SETS..BRANCH
:27801 : NAD/GROUP1 ; TO NEXT WORD AND THEN BRANCH ON THE
:27802 : ; FUNCTION BITS
:27803 -----
:27804 :186:
:27805 :=1**
:27806 :
:27807 : BEN0/F0,
:27808 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:27809 : BEN2/F2,
:27810 : NAD/GROUP1.F0

```

ZZ-ENKCF-1.4  
: ENKCF.MCR  
: ENKCF.MIC

ENKCF.MIC

MICRO2 1M(01)

Test 31 DAR SERIAL15-MAR-83

15-MAR-83 11:46:22

K 12

Fiche 3 Frame K12

Sequence 565

Rev 01.40

Page 559

ZZ-E

:

```
:27811 :-----
:27812 :028:
:27813 :=000
:27814 :GROUP1.F0:
:27815 :
:27816 :CLR.MATCH.1:
:27817 :
:27818 : UCON/CLR.MATCH, ;CLR THE MATCH CONDITION
:27819 : NAD/TEST.MATCH.1
:27820 :-----
:27821 :1F6:
:27822 :TEST.MATCH.1:
:27823 :
:27824 : BEN2/MISMATCH.L, ;IS MATCH ASSERTED?
:27825 : NAD/BEN2.1
:27826 :-----
:27827 :1A0:
:27828 :=0**
:27829 :BEN2.1:
:27830 :
:27831 : INCR.DAR/ON, ;INCR DAR
:27832 : NAD/XFER.REQ ;FLAG THE CPU
:27833 :-----
:27834 :1A4:
:27835 :=1**
:27836 :BEN2.2:
:27837 :
:27838 : INCR.DAR/ON, ;INCR DAR IF BEN2 CONDITION ASSERTED
:27839 : NAD/BEN2.1 ;INCR DAR AGAIN IF BEN2 COND ASSERTED
:27840 :-----
:27841 :1D8:
:27842 :XFER.REQ:
:27843 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:27844 : NAD/DIAG.WAIT
:27845 :-----
:27846 :1D7:
:27847 :DIAG.WAIT:
:27848 :
:27849 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:27850 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:27851 :-----
:27852 :159:
:27853 :=*0*
:27854 :DIAG.WAIT1:
:27855 :
:27856 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:27857 : ; COMMAND FROM THE CPU
:27858 :-----
:27859 :15B:
:27860 :=*1*
:27861 :DIAG.WAIT2:
:27862 :
:27863 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:27864 : ; WAIT SOME MORE
:27865 :-----
```

U 17  
U 17  
U 17

U 17  
U 17  
U 17

U 17  
U 17  
U 17  
U 17  
U 17

U 17  
U 17

U 17  
U 17

U 17  
U 17  
U 17  
U 17  
U 17  
U 17  
U 17

```

:27866 WAIT.IDC.T31.1:
U 1757, DB00,1A :27867 SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ FROM IDC
U 1758, 8975,A4 :27868 JMP [IDC.DONE.T31.1] ;XFER REQ SET
U 1759, 0975,74 :27869 JMP [WAIT.IDC.T31.1] ;WAIT SOME MORE
:27870
:27871 IDC.DONE.T31.1:
U 175A, B700,15 :27872 MOV LS[SETCSR.F0] TO WR[0] ;CLEAR CSR F<2:0>
U 175B, 17A0,15 :27873 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
U 175C, 90FA,15 :27874 MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
:27875
U 175D, A006,15 :27876 MOV WR[3] TO WR[0] ;GET THE DAR PATTERN TO WRITE
U 175E, 97A4,15 :27877 PORT.WRITE PORT.ADRS[DAR] WITH WR[0] ;WRITE THE DAR
:27878
U 175F, B610,15 :27879 MOV LS[T8] TO WR[0] ;GET THE DATA TO WRITE
U 1760, 17D8,15 :27880 PORT.CONTROL [SEL.FIFO.A] ;SELECT FIFO A
U 1761, 17C0,15 :27881 PORT.CONTROL [CLEAR.FIFO.CNTR] ;SET FIFO A ADRS=0
U 1762, 97A8,15 :27882 PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;WRITE THE DATA TO FIFO A
:27883
U 1763, 17C0,15 :27884 PORT.CONTROL [CLEAR.FIFO.CNTR] ;SET FIFO A ADRS=0
:27885
:27886 ;SET UP THE CSR TO BRANCH TO THE CORRECT IDC ROUTINE
:27887
U 1764, 370E,15 :27888 MOV LS[SETCSR.F7] TO WR[0]
U 1765, 17A0,15 :27889 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:27890
:27891 MOV LS[SETCSR.R80] TO WR[0] ;MAINT=0,CRDY=0,F<2:0>=111
U 1766, 379E,15 :27892 MOV LS[SETCSR.F2] TO WR[1] ;SET UP THE CSR TO CONTINUE TO
U 1767, B704,95 :27893 BIS WR[1] TO WR[0] ;BRANCH TO THE CORRECT ROUTINE
U 1768, AEC2,15 :27894 BIS WR[2] TO WR[0]
U 1769, AEC4,15 :27895 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;INCLUDE THE RLO2 DRV SELECT
U 176A, 17A0,15 :27896 ;WRITE THE CSR
:27897
:27898 -----
:27899 *****
:27900 * IDC *
:27901 *****
:27902 -----
:27903 :018:
:27904 :=000
:27905 :DIAG.IDLE:
:27906
:27907 BEN0/F0,
:27908 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:27909 BEN2/F2,
:27910 NAD/DIAG.IDLE
:27911 -----
:27912 :01F:
:27913 :=111
:27914
:27915 BEN0/CRDY.L,
:27916 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:27917 NAD/TEST.FUNC7 ;DEPENDENT ON CSR <CRDY> AND <MAINT>
:27918 -----
:27919 :079:
:27920 :=0*1

```

```

:27921 :
:27922 : NAD/GROUP1
:27923 :-----
:27924 :182:
:27925 :=0**
:27926 :GROUP1:
:27927 :
:27928 : BEN2/R80.FORMAT.H, ;WHEN THE R80 FORMAT BIT SETS..BRANCH
:27929 : NAD/GROUP1 ; TO NEXT WORD AND THEN BRANCH ON THE
:27930 : ; FUNCTION BITS
:27931 :-----
:27932 :186:
:27933 :=1**
:27934 :
:27935 : BEN0/F0,
:27936 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:27937 : BEN2/F2,
:27938 : NAD/GROUP1.F0
:27939 :-----
:27940 :02A:
:27941 :=010
:27942 :MATCH.OR.WRTCHK.1:
:27943 :
:27944 : UMAINT/ON,
:27945 : LOOP.LOCK/ON, ;MAINT PREVENTS LOOP LOCK FROM SELECTING
:27946 : DISK.WRITE/ON, ;READ DATA...DISK WRITE STILL SELECTS DSRO
:27947 : LOAD.LOOP.CNTR.[OE0],
:27948 : NAD/MATCH.OR.WRTCHK.2
:27949 :-----
:27950 :198:
:27951 :=0
:27952 :MATCH.OR.WRTCHK.2:
:27953 :
:27954 : UMAINT/ON,
:27955 : LOOP.LOCK/ON, ;SHIFT OUT 32 ZEROS TO SYNC UP
:27956 : DISK.WRITE/ON, ; DATA SEPARATOR
:27957 : UCMD/SET.FIFOA,
:27958 : INCR.LPCNTR/ON,
:27959 : BEN0/CNTR.OVFLW,
:27960 : NAD/MATCH.OR.WRTCHK.2
:27961 :-----
:27962 :199:
:27963 :=1
:27964 :MATCH.OR.WRTCHK.3:
:27965 :
:27966 : UMAINT/ON,
:27967 : LOOP.LOCK/ON,
:27968 : DISK.WRITE/ON, ;MUST WAIT 3 CYCLES FOR SYNC SEEN
:27969 : ENB.DSK.CLK,
:27970 : ENB.CLR.BC/ON, ;CLR THE BIT COUNTER WHEN SYNC SEEN
:27971 : NAD/MATCH.OR.WRTCHK.4 ; FOR THE MATCH TEST
:27972 :-----
:27973 :1F2:
:27974 :MATCH.OR.WRTCHK.4:
:27975 :

```



```

:27976 : UMAINT/ON,
:27977 : LOOP.LOCK/ON, ;WAIT SOME MORE
:27978 : DISK.WRITE/ON,
:27979 : ENB.CLR.BC/ON, ;CLR THE BIT COUNTER WHEN SYNC SEEN
:27980 : NAD/MATCH.OR.WRTCHK.5 ; FOR THE MATCH TEST
:27981 :-----
:27982 :1F3:
:27983 :MATCH.OR.WRTCHK.5:
:27984 :
:27985 : UMAINT/ON,
:27986 : LOOP.LOCK/ON, ;WAIT SOME MORE
:27987 : DISK.WRITE/ON,
:27988 : ENB.CLR.BC/ON, ;CLR THE BIT COUNTER WHEN SYNC SEEN
:27989 : NAD/MATCH.OR.WRTCHK.6 ; FOR THE MATCH TEST
:27990 :-----
:27991 :1F4:
:27992 :MATCH.OR.WRTCHK.6:
:27993 :
:27994 : UMAINT/ON,
:27995 : LOOP.LOCK/ON,
:27996 : DISK.WRITE/ON, ;LOAD THE DATA FROM FIFO A INTO THE DSR
:27997 : ENB.FIFO/ON,
:27998 : LOAD.DSR/ON,
:27999 : BEN2/MAINT, ;MATCH TEST OR WRTCHK TEST?
:28000 : NAD/MATCH.OR.WRTCHK.7
:28001 :-----
:28002 :183:
:28003 :=0**
:28004 :MATCH.OR.WRTCHK.7:
:28005 :
:28006 : UMAINT/ON,
:28007 : LOOP.LOCK/ON, ;MATCH TEST
:28008 : DISK.WRITE/ON,
:28009 : LOAD.LOOP.CNTR.[OF6],
:28010 : NAD/MATCH.1
:28011 :-----
:28012 :19A:
:28013 :=0
:28014 :MATCH.1:
:28015 :
:28016 : UMAINT/ON,
:28017 : LOOP.LOCK/ON,
:28018 : DISK.WRITE/ON, ;WAIT FOR DATA TO SHIFT THRU DISKLESS
:28019 : INCR.LPCNTR/ON, ; LOOP TO BECOME RD DATA
:28020 : BEN0/CNTR.OVFLW,
:28021 : NAD/MATCH.1
:28022 :-----
:28023 :19B:
:28024 :=1
:28025 :MATCH.2:
:28026 :
:28027 : UMAINT/ON, ;KEEP THE CLK FROM THE DATA SEPARATOR
:28028 : LOOP.LOCK/ON, ; GOING AND TEST MATCH
:28029 : DISK.WRITE/ON,
:28030 : UCON/TEST.MATCH,

```

```
:28031 : NAD/MATCH.3
:28032 : -----
:28033 : 1F5:
:28034 : MATCH.3:
:28035 :
:28036 : UMAINT/ON, ;KEEP THE CLK FROM THE DATA SEPARATOR
:28037 : LOOP.LOCK/ON, ; GOING
:28038 : DISK.WRITE/ON,
:28039 : ENB.CPU.CLK, ;SELECT THE CPU CLK
:28040 : NAD/TEST.MATCH.1
:28041 : -----
:28042 : 1F6:
:28043 : TEST.MATCH.1:
:28044 :
:28045 : BEN2/MISMATCH.L, ;IS MATCH ASSERTED?
:28046 : NAD/BEN2.1
:28047 : -----
:28048 : 1A0:
:28049 : =0**
:28050 : BEN2.1:
:28051 :
:28052 : INCR.DAR/ON, ;INCR DAR
:28053 : NAD/XFER.REQ ;FLAG THE CPU
:28054 : -----
:28055 : 1A4:
:28056 : =1**
:28057 : BEN2.2:
:28058 :
:28059 : INCR.DAR/ON, ;INCR DAR IF BEN2 CONDITION ASSERTED
:28060 : NAD/BEN2.1 ;INCR DAR AGAIN IF BEN2 COND ASSERTED
:28061 : -----
:28062 : 1D8:
:28063 : XFER.REQ:
:28064 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:28065 : NAD/DIAG.WAIT
:28066 : -----
:28067 : 1D7:
:28068 : DIAG.WAIT:
:28069 :
:28070 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:28071 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:28072 : -----
:28073 : 159:
:28074 : =*0*
:28075 : DIAG.WAIT1:
:28076 :
:28077 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:28078 : ; COMMAND FROM THE CPU
:28079 : -----
:28080 : 158:
:28081 : =*1*
:28082 : DIAG.WAIT2:
:28083 :
:28084 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:28085 : ; WAIT SOME MORE
```

```

:28086 :-----
:28087 WAIT.IDC.T31.2:
U 176B, DB00,1A :28088 SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ FROM IDC
U 176C, 0976,E4 :28089 JMP [IDC.DONE.T31.2] ;XFER REQ SET
U 176D, 0976,B4 :28090 JMP [WAIT.IDC.T31.2] ;WAIT SOME MORE
:28091
:28092 IDC.DONE.T31.2:
U 176E, B700,15 :28093 MOV LS[SETCSR.F0] TO WR[0] ;CLEAR CSR F<2:0>
U 176F, 17A0,15 :28094 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
U 1770, 90FA,15 :28095 MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
:28096
:28097 READ.DAR.T31.1:
U 1771, 9090,15 :28098 MISC [SET.PORT.I.0] ;SELECT THE IDC
U 1772, 1784,15 :28099 PORT.READ PORT.ADRS[DAR] ;SET UP TO READ THE DAR
U 1773, 3B32,15 :28100 MOV PORT TO LS[PORT0] ;READ THE DATA FROM DAR AND
:28101 ; SAVE IT IN LS
U 1774, 3732,15 :28102 MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
U 1775, 2006,95 :28103 MOV WR[3] TO WR[1] ;SET UP EXPECTED DATA
U 1776, C742,95 :28104 BIS LS[BIT1] TO WR[1]
U 1777, 0869,3C :28105 JSR [CHECK.RESULT] ;CHECK THE DATA
U 1778, 0975,14 :28106 JMP [LOOP.T31.1] ;LOOP HERE IF ERR & LOE IS SET
:28107
:28108
:28109 BIT LS[BIT15] WITH WR[3], ;DONE?
U 1779, 595F,B5 :28110 DT(LONG)&SET.ALU.CC
U 177A, 0978,01 :28111 JMP.IF[BIT.SET] TO [CLEANUP.T31] ;JMP IF DONE
:28112
:28113 ;GET NEXT DATA PATTERN FOR THE DAR
:28114
U 177B, A3C1,95 :28115 ROL WR[3] ;GET THE LAST DAR PATTERN
:28116 ;ROTATE ONCE LEFT
:28117 ;GET THE NEXT FIFO PATTERN
:28118
U 177C, B610,15 :28119 MOV LS[T8] TO WR[0] ;GET THE LAST FIFO PATTERN
U 177D, A3C0,15 :28120 ROL WR[0] ;ROTATE ONCE LEFT
U 177E, 3E10,15 :28121 MOV WR[0] TO LS[T8] ;SAVE IN LS[T8]
:28122
U 177F, 0975,14 :28123 JMP [TST.NXT.PAT.T31] ;GO TEST NEXT PATTERN
:28124
:28125 CLEANUP.T31:
U 1780, 370E,15 :28126 MOV LS[SETCSR.F7] TO WR[0] ;WRITE THE CSR
U 1781, 17A0,15 :28127 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;MAINT=0,CRDY=0,F<2:0>=111
:28128 ;SET UP THE CSR TO CONTINUE TO
:28129 ; BRANCH TO THE CORRECT ROUTINE
U 1782, 379E,15 :28129 MOV LS[SETCSR.R80] TO WR[0]
U 1783, 3700,95 :28130 MOV LS[SETCSR.F0] TO WR[1]
U 1784, AEC2,15 :28131 BIS WR[1] TO WR[0]
U 1785, 17A0,15 :28132 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:28133 ;F<2:0> = 000
:28134 ;UCLR MATCH
:28135 :-----
:28136 :-----
:28137 *****
:28138 * IDC *
:28139 *****
:28140 :-----

```

```

:28141 :018:
:28142 :=000
:28143 :DIAG.IDLE:
:28144 :
:28145 : BEN0/F0,
:28146 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:28147 : BEN2/F2,
:28148 : NAD/DIAG.IDLE
:28149 :-----
:28150 :01F:
:28151 :=111
:28152 :
:28153 : BEN0/CRDY.L,
:28154 : BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:28155 : NAD/TEST.FUNC7 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:28156 :-----
:28157 :079:
:28158 :=0*1
:28159 :
:28160 : NAD/GROUP1
:28161 :-----
:28162 :182:
:28163 :=0**
:28164 :GROUP1:
:28165 :
:28166 : BEN2/R80.FORMAT.H, ;WHEN THE R80 FORMAT BIT SETS..BRANCH
:28167 : NAD/GROUP1 ; TO NEXT WORD AND THEN BRANCH ON THE
:28168 : ; FUNCTION BITS
:28169 :-----
:28170 :186:
:28171 :=1**
:28172 :
:28173 : BEN0/F0,
:28174 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:28175 : BEN2/F2,
:28176 : NAD/GROUP1.F0
:28177 :-----
:28178 :028:
:28179 :=000
:28180 :GROUP1.F0:
:28181 :
:28182 :CLR.MATCH.1:
:28183 :
:28184 : UCON/CLR.MATCH, ;CLR THE MATCH CONDITION
:28185 : NAD/TEST.MATCH.1
:28186 :-----
:28187 :1F6:
:28188 :TEST.MATCH.1:
:28189 :
:28190 : BEN2/MISMATCH.L, ;IS MATCH ASSERTED?
:28191 : NAD/BEN2.1
:28192 :-----
:28193 :1A0:
:28194 :=0**
:28195 :BEN2.1:

```

```

:28196 :
:28197 : INCR.DAR/ON, ;INCR DAR
:28198 : NAD/XFER.REQ ;FLAG THE CPU
:28199 :-----
:28200 :1A4:
:28201 :=1**
:28202 :BEN2.2:
:28203 :
:28204 : INCR.DAR/ON, ;INCR DAR IF BEN2 CONDITION ASSERTED
:28205 : NAD/BEN2.1 ;INCR DAR AGAIN IF BEN2 COND ASSERTED
:28206 :-----
:28207 :1D8:
:28208 :XFER.REQ:
:28209 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:28210 : NAD/DIAG.WAIT
:28211 :-----
:28212 :1D7:
:28213 :DIAG.WAIT:
:28214 :
:28215 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:28216 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:28217 :-----
:28218 :159:
:28219 :=*0*
:28220 :DIAG.WAIT1:
:28221 :
:28222 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:28223 : ; COMMAND FROM THE CPU
:28224 :-----
:28225 :15B:
:28226 :=*1*
:28227 :DIAG.WAIT2:
:28228 :
:28229 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:28230 : ; WAIT SOME MORE
:28231 :-----
:28232 :WAIT.IDC.T31.3:
:28233 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ FROM IDC
:28234 : JMP [IDC.DONE.T31.3] ;XFER REQ SET
:28235 : JMP [WAIT.IDC.T31.3] ;WAIT SOME MORE
:28236 :
:28237 :IDC.DONE.T31.3:
:28238 : MOV LS[SETCSR.F0] TO WR[0] ;CLEAR CSR F<2:0>
:28239 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:28240 : MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
:28241 :
:28242 :
:28243 : JSR [CLEANUP] ;CLEANUP THE CONDITIONS SET BY
:28244 : ; THIS TEST
:28245 : END.T31: ;END TEST
:28246 :
:28247 :
:28248 :
:28249 :

```

U 1786, DB00,1A  
 U 1787, 0978,94  
 U 1788, 0978,64

U 1789, B700,15  
 U 178A, 17A0,15  
 U 178B, 90FA,15

U 178C, 8875,3C

ZZ-  
 :  
 :

U 1  
 U 1  
 U 1

U 1  
 U 1  
 U 1

U 1

:28250 :PAGE "Test 32 WRITE CHECK TEST (M8388 IDC MODULE)"

:28251 :++

:28252 :

:28253 :Test Description:

:28254 :

:28255 :This test is designed to insure that the data compares correctly on  
:28256 :a write check operation. The data will be loaded into the FIFO  
:28257 :and shifted through the diskless loop to become READ DATA. The data  
:28258 :to be compared will be loaded into the Data Shifted Register  
:28259 :and shifted out as DSR 0 and will be compared with the READ DATA in the  
:28260 :DSK RD SEQ REG PAL. A TEST MATCH will then be executed with  
:28261 :BEN2/MISMATCH L selected. If a no match is detected, the DAR will  
:28262 :be incremented once. If a Match is detected then the DAR will be  
:28263 :incremented twice. The CPU will then check the DAR.

:28264 :

:28265 :Data Patterns (2 bits):

:28266 :

|            | READ DATA | DSR |
|------------|-----------|-----|
| Pattern 1: | 01        | 01  |
| Pattern 2: | 01        | 00  |
| Pattern 3: | 00        | 01  |

:28270 :

:28271 :Assumptions:

:28272 :

:28273 :It is assumed that all previous tests have run successfully.

:28274 :

:28275 :Test Steps:

:28276 :

- :28277 :1. Set up error mask, error number and module number in LS  
:28278 : (for error printouts) and clear the previous error number in LS.
- :28279 :2. Force the IDC microcode to DIAG.IDLE.
- :28280 :3. If the drive sizing routine has not been run previously, JSR to  
:28281 : DRIVE.SIZE.
- :28282 :4. Select an RL02 drive.
- :28283 :5. Load the data pattern into the FIFO.
- :28284 :6. Clear the DAR.
- :28285 :7. Set up the CSR to branch to the correct IDC microcode routine.

:28286 :

:28287 :

:28288 :

:28289 :\*\*\*\*\*

:28290 :\* IDC \*

:28291 :\*\*\*\*\*

:28292 :

- :28293 :IDC1. Select the disk clock.
- :28294 :IDC2. Shift out the sync character.
- :28295 :IDC3. Follow with the first data from the FIFO and  
:28296 : shift it through the diskless loop to become READ DATA.
- :28297 :IDC4. Load the next part of the data pattern into the DSR.
- :28298 :IDC5. Execute an instruction with UCON <1:0> = 10, TEST MATCH.
- :28299 :IDC6. Select the CPU clock.
- :28300 :IDC7. Select BEN2/MISMATCH L.
- :28301 :IDC8. If MISMATCH L is asserted then increment the DAR once.  
:28302 : If MISMATCH L is not asserted (a match was detected) then  
:28303 : increment DAR twice.
- :28304 :IDC9. Issue XFER.REQ to flag the CPU.

```

:28305 :
:28306 : 8. Wait for a XFER.REQ from IDC.
:28307 : 9. Clear CSR <2:0> and issue XFER GRANT to clear the XFER.REQ.
:28308 : 10. Check the contents of the DAR.
:28309 : 11. Get the next data pattern and go to Step 5 until all patterns have
:28310 : been tested.
:28311 : 12. Clear IDC.
:28312 : 13. End test.
:28313 :
:28314 : Errors:
:28315 :
:28316 : ERROR 1 - Data did not compare correctly on a write check operation.
:28317 :
:28318 :
:28319 : Printout:
:28320 : EXPECTED RECEIVED OTHER DATA
:28321 : DAR DAR
:28322 :
:28323 : BYTE 0 = RD DATA
:28324 : BYTE 1 = DSR DATA
:28325 :
:28326 : Debug:
:28327 : ERROR 1 -
:28328 : The data pattern is loaded into FIFO A address 0. Byte 0 is the data
:28329 : that will be shifted through the RL02 diskless loop to become
:28330 : M.READ DATA H and BYTE 1 is the data loaded into the DSR and then
:28331 : shifted out through DAR 0 which is an input to the DSK RD SEQ REG PAL.
:28332 : A line with no disk drive attached or with an RL02 drive attached is
:28333 : selected and will assert RL02 H which is inverted to become R80 H = L.
:28334 : The IDC will branch to a routine that will load the Byte 0 of FIFO A
:28335 : address 0, into the DSR. This data is then shifted through the RL02
:28336 : diskless loop to become READ DATA H (see RL02 DATA PATH TESTS if
:28337 : diskless loop is not functioning correctly).
:28338 : READ DATA H is an input to the SERIAL DAR LO PAL and it becomes
:28339 : M.READ DATA H. This signal is an input to the DSK RD SEQ REG PAL.
:28340 : As bit 0 of this data becomes M.READ DATA H, the DSR is loaded with
:28341 : Byte 1 of FIFO A data.
:28342 : At the DSK RD SEQ REG PAL, SYNC SEEN L will be asserted at the next
:28343 : CURRENT CLK H after SEL DSK CLK H = H and BYTE COMPARE H = H. After
:28344 : SYNC SEEN L is asserted, COMPARE L will assert when DSR 0 =
:28345 : M.READ DATA H with UWRT CHK H = H. COMPARE L will assert on the first
:28346 : match of DSR 0 H and M.READ DATA H and will remain asserted
:28347 : unless a mismatch is encountered. After SYNC SEEN L, M.READ DATA H and
:28348 : DSR 0 H should be zeros until the byte of data has shifted through the
:28349 : diskless loop and becomes M.READ DATA H and the DSR is loaded again.
:28350 : The IDC will then issue an instruction with UCON <1:0> = 10. These bits
:28351 : will be decoded by the DSK RD SEQ REG PAL to perform a TEST.MATCH.
:28352 : If COMPARE L is asserted at this time, MISMATCH L = H (data pat 1).
:28353 : If COMPARE L is not asserted at this time, MISMATCH L = L
:28354 : (data pat 2 and 3).
:28355 : The IDC microcode will then select BEN2/MISMATCH L (BEN2 S2 H = L,
:28356 : BEN2 S1 H = L, BEN2 S0 H = H). If MISMATCH L is not asserted,
:28357 : (data pat 1), the microcode will branch to a series of instructions
:28358 : that will increment the DAR twice.
:28359 : If MISMATCH L is asserted (data pat 2 and 3), the microcode will branch

```

```

:28360 : to an instruction that will increment the DAR once.
:28361 :--
:28362 :*****
:28363 T.32:
U 178D, B65E,15 :28364 MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT 15 IN WR[0] FOR
:28365 ; CONTROL AND STATUS WORD
U 178E, 3E80,15 :28366 MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND
:28367 ; STATUS WORD - BIT 15
:28368 ; INDICATES START OF TEST TO
:28369 ; CONSOLE
U 178F, 10E0,15 :28370 MISC [SET.CP.ATTN] ;ISSUE CPU ATTN TO CONSOLE
:28371 WAIT.T32.0:
U 1790, 8979,04 :28372 JMP [WAIT.T32.0] ;LOOP HERE WAITING FOR
:28373 ; CONSOLE TO RESPOND
:28374
U 1791, 65A0,15 :28375 CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
:28376 ; IN LS
:28377
U 1792, B64A,15 :28378 MOV LS[IDC] TO WR[0] ;STORE MODULE CODE IN LS TO
U 1793, 3E8C,15 :28379 MOV WR[0] TO LS[MODULE.NUM] ; INDICATE IDC MODULE
:28380
:28381
U 1794, B640,15 :28382 MOV LS[#1] TO WR[0] ;SET WRO = 1
U 1795, BE82,15 :28383 MOV WR[0] TO LS[ERROR.NUMBER] ;SET UP ERROR NUMBER = 1 IN LS
:28384
U 1796, 3776,15 :28385 MOV LS[DAR.MASK] TO WR[0] ;GET THE MASK TO CHECK THE
U 1797, 3E8A,15 :28386 MOV WR[0] TO LS[ERROR.MASK] ; DAR
:28387
U 1798, B670,15 :28388 MOV LS[OTHER.DATA] TO WR[0] ;SET UP TO PRINT OTHER DATA
U 1799, 3E80,15 :28389 MOV WR[0] TO LS[CONTROL.STATUS] ; IF ERROR
:28390
U 179A, B78E,15 :28391 MOV LS[DRIVE.STATUS] TO WR[0] ;HAS THE DRIVE SIZING ROUTINE
:28392 BIT LS[BIT7] WITH WR[0], ; BEEN RUN?
U 179B, D94E,35 :28393 DT(LONG)&SET.ALU.CC
U 179C, 0979,E1 :28394 JMP.IF[BIT.SET] TO [SIZE.OK.T32.1] ;IF YES THEN GO RUN TEST
:28395
U 179D, 8870,0C :28396 JSR [DRIVE.SIZE] ;GO RUN THE SIZING ROUTINE
:28397
:28398 SIZE.OK.T32.1:
U 179E, 8872,EC :28399 JSR [DIAG.CODE] ;FORCE THE IDC TO DIAG.IDLE
:28400
:28401 ;SELECT AN RL02 DRIVE AND SAVE THE INFO IN WR[2]
:28402
U 179F, B78E,15 :28403 MOV LS[DRIVE.STATUS] TO WR[0]
:28404 BIT LS[BIT0] WITH WR[0], ;IS DRIVE 0 AN R80?
U 17A0, 5940,35 :28405 DT(LONG)&SET.ALU.CC
U 17A1, 897A,49 :28406 JMP.IF[BITS.CLR] TO [SET.DS0.T32.1] ;JMP IF DRV 0 IS NOT AN R80
U 17A2, B717,15 :28407 MOV LS[SETCSR.DS1] TO WR[2] ;SAVE THE DATA TO SEL DRIVE 1
U 17A3, 897A,54 :28408 JMP [TEST.T32.1]
:28409 SET.DS0.T32.1:
U 17A4, 3715,15 :28410 MOV LS[SETCSR.DS0] TO WR[2] ;SAVE THE DATA TO SEL DRIVE 0
:28411
:28412 TEST.T32.1:
:28413 ;SET UP THE DATA IN FIFO A (FIRST BYTE WILL CONTAIN THE READ DATA PATTERN
:28414 ;AND THE SECOND BYTE WILL CONTAIN THE DSR DATA PATTERN)

```



```

:28415
U 17A5, 37BB,95 :28416 MOV LS[#101] TO WR[3] ;GET THE FIRST PATTERN
U 17A6, 3642,15 :28417 MOV LS[#2] TO WR[0] ;SET UP THE EXPECTED DATA
U 17A7, 3E10,15 :28418 MOV WR[0] TO LS[T8] ;SAV EXP DATA IN LS[T8]
U 17A8, 3642,15 :28419 MOV LS[#2] TO WR[0]
U 17A9, BE12,15 :28420 MOV WR[0] TO LS[T9] ;SET UP THE LOOP CNTR
:28421
:28422 TST.NXT.PAT.T32:
:28423 LOOP.T32.1:
U 17AA, 370E,15 :28424 MOV LS[SETCSR.F7] TO WR[0]
U 17AB, 17A0,15 :28425 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:28426 ;MAINT=0,CRDY=0,F<2:0>=111
:28427 ;CONTINUE TO BRANCH
U 17AC, 379E,15 :28428 MOV LS[SETCSR.R80] TO WR[0] ;SET UP THE CSR TO CONTINUE TO
U 17AD, 3700,95 :28429 MOV LS[SETCSR.F0] TO WR[1] ; BRANCH TO THE CORRECT ROUTINE
U 17AE, AEC2,15 :28430 BIS WR[1] TO WR[0]
U 17AF, 17A0,15 :28431 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:28432 ;F<2:0> = 000
:28433 ;UCLR MATCH
:28434 -----
:28435 -----
:28436 ;*****
:28437 ; IDC *
:28438 ;*****
:28439 -----
:28440 ;018:
:28441 ;=000
:28442 ;DIAG.IDLE:
:28443 ;
:28444 ; BEN0/F0,
:28445 ; BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:28446 ; BEN2/F2,
:28447 ; NAD/DIAG.IDLE
:28448 ;-----
:28449 ;01F:
:28450 ;=111
:28451 ;
:28452 ; BEN0/CRDY.L,
:28453 ; BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:28454 ; NAD/TEST.FUNC7 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:28455 ;-----
:28456 ;079:
:28457 ;=0*1
:28458 ;
:28459 ; NAD/GROUP1
:28460 ;-----
:28461 ;182:
:28462 ;=0**
:28463 ;GROUP1:
:28464 ;
:28465 ; BEN2/R80.FORMAT.H, ;WHEN THE R80 FORMAT BIT SETS..BRANCH
:28466 ; NAD/GROUP1 ; TO NEXT WORD AND THEN BRANCH ON THE
:28467 ; ; FUNCTION BITS
:28468 ;-----
:28469 ;186:

```

[illegible]

ZZ-ENKCF-1.4 ;  
: ENKCF.MCR  
: ENKCF.MIC

ENKCF.MIC

J 13  
Test 32 WRITE CHECK15-MAR-83  
MICRO2 1M(01) 15-MAR-83 11:46:22  
Test 32 WRITE CHECK TEST (M8388 IDC MODULE)

Fiche 3 Frame J13

Sequence 577

Rev 01.40 Page 571

ZZ-1  
:  
:

```
:28470 : =1**
:28471 :
:28472 : BEN0/F0,
:28473 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:28474 : BEN2/F2,
:28475 : NAD/GROUP1.F0
:28476 : -----
:28477 : 028:
:28478 : =000
:28479 : GROUP1.F0:
:28480 :
:28481 : CLR.MATCH.1:
:28482 :
:28483 : UCON/CLR.MATCH, ;CLR THE MATCH CONDITION
:28484 : NAD/TEST.MATCH.1
:28485 : -----
:28486 : 1F6:
:28487 : TEST.MATCH.1:
:28488 :
:28489 : BEN2/MISMATCH.L, ;IS MATCH ASSERTED?
:28490 : NAD/BEN2.1
:28491 : -----
:28492 : 1A0:
:28493 : =0**
:28494 : BEN2.1:
:28495 :
:28496 : INCR.DAR/ON, ;INCR DAR
:28497 : NAD/XFER.REQ ;FLAG THE CPU
:28498 : -----
:28499 : 1A4:
:28500 : =1**
:28501 : BEN2.2:
:28502 :
:28503 : INCR.DAR/ON, ;INCR DAR IF BEN2 CONDITION ASSERTED
:28504 : NAD/BEN2.1 ;INCR DAR AGAIN IF BEN2 COND ASSERTED
:28505 : -----
:28506 : 1D8:
:28507 : XFER.REQ:
:28508 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:28509 : NAD/DIAG.WAIT
:28510 : -----
:28511 : 1D7:
:28512 : DIAG.WAIT:
:28513 :
:28514 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:28515 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:28516 : -----
:28517 : 159:
:28518 : =*0*
:28519 : DIAG.WAIT1:
:28520 :
:28521 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:28522 : ; COMMAND FROM THE CPU
:28523 : -----
:28524 : 15B:
```

[illegible]

[illegible]

```

:28635 :
:28636 : UMAINT/ON,
:28637 : LOOP.LOCK/ON,
:28638 : DISK.WRITE/ON, ;MUST WAIT 3 CYCLES FOR SYNC SEEN
:28639 : ENB.DSK.CLK,
:28640 : ENB.CLR.BC/ON, ;CLR THE BIT COUNTER WHEN SYNC SEEN
:28641 : NAD/MATCH.OR.WRTCHK.4 ; FOR THE MATCH TEST
:28642 :-----
:28643 :1F2:
:28644 :MATCH.OR.WRTCHK.4:
:28645 :
:28646 : UMAINT/ON,
:28647 : LOOP.LOCK/ON, ;WAIT SOME MORE
:28648 : DISK.WRITE/ON,
:28649 : ENB.CLR.BC/ON, ;CLR THE BIT COUNTER WHEN SYNC SEEN
:28650 : NAD/MATCH.OR.WRTCHK.5 ; FOR THE MATCH TEST
:28651 :-----
:28652 :1F3:
:28653 :MATCH.OR.WRTCHK.5:
:28654 :
:28655 : UMAINT/ON,
:28656 : LOOP.LOCK/ON, ;WAIT SOME MORE
:28657 : DISK.WRITE/ON,
:28658 : ENB.CLR.BC/ON, ;CLR THE BIT COUNTER WHEN SYNC SEEN
:28659 : NAD/MATCH.OR.WRTCHK.6 ; FOR THE MATCH TEST
:28660 :-----
:28661 :1F4:
:28662 :MATCH.OR.WRTCHK.6:
:28663 :
:28664 : UMAINT/ON,
:28665 : LOOP.LOCK/ON,
:28666 : DISK.WRITE/ON, ;LOAD THE DATA FROM FIFO A INTO THE DSR
:28667 : ENB.FIFO/ON, ;PREVIOUS CODE IS USED FOR MATCH TESTS AND
:28668 : LOAD.DSR/ON, ;WRT CHK TESTS, SO NOW USE DIFFERNT CODE FOR
:28669 : ; THE REST OF EACH TEST
:28670 : BEN2/MAINT, ;MATCH TEST OR WRTCHK TEST?
:28671 : NAD/MATCH.OR.WRTCHK.7
:28672 :-----
:28673 :187:
:28674 :=1**
:28675 :MATCH.OR.WRTCHK.8:
:28676 : UMAINT/ON,
:28677 : LOOP.LOCK/ON,
:28678 : DISK.WRITE/ON, ;WRITE CHECK TEST
:28679 : LOAD.LOOP.CNTR.[0FE],
:28680 : INCR.FIFO.CNTR/ON, ;SET ADRS TO NEXT BYTE OF DATA
:28681 : NAD/WRT.CHK.1
:28682 :-----
:28683 :19C:
:28684 :=0
:28685 :WRT.CHK.1:
:28686 :
:28687 : UMAINT/ON,
:28688 : LOOP.LOCK/ON,
:28689 : DISK.WRITE/ON, ;WAIT FOR DATA TO SHIFT THRU DISKLESS

```

ZZ-6  
 :  
 :  
 U 18  
 U 18  
 U 18  
 U 18  
 U 18

```
:28690 : INCR.LPCNTR/ON, ; LOOP TO BECOME RD DATA
:28691 : BEN0/CNTR.OVFLW,
:28692 : WRT.CHK/ON,
:28693 : NAD/WRT.CHK.1
:-----
:28694 :
:28695 : 19D:
:28696 : =1
:28697 : WRT.CHK.2:
:28698 : UMAINT/ON,
:28699 : LOOP.LOCK/ON,
:2700 : DISK.WRITE/ON, ;LOAD THE SECOND BYTE OF THE DATA FROM FIFO A
:2701 : ENB.FIFO/ON, ; INTO THE DSR
:2702 : LOAD.DSR/ON,
:2703 : WRT.CHK/ON,
:2704 : NAD/WRT.CHK.3
:-----
:2705 :
:2706 : 1F7:
:2707 : WRT.CHK.3:
:2708 : UMAINT/ON, ;KEEP THE CLK FROM THE DATA SEPARATOR
:2709 : LOOP.LOCK/ON, ; GOING
:2710 : DISK.WRITE/ON,
:2711 : WRT.CHK/ON,
:2712 : NAD/WRT.CHK.4
:-----
:2713 :
:2714 : 1F8:
:2715 : WRT.CHK.4:
:2716 : UMAINT/ON, ;KEEP THE CLK FROM THE DATA SEPARATOR
:2717 : LOOP.LOCK/ON, ; GOING AND TEST MATCH
:2718 : DISK.WRITE/ON,
:2719 : UCON/TEST.MATCH,
:2720 : WRT.CHK/ON,
:2721 : NAD/WRT.CHK.5
:-----
:2722 :
:2723 : 1F9:
:2724 : WRT.CHK.5:
:2725 : UMAINT/ON, ;KEEP THE CLK FROM THE DATA SEPARATOR
:2726 : LOOP.LOCK/ON,
:2727 : DISK.WRITE/ON,
:2728 : ENB.CPU.CLK, ;SELECT THE CPU CLK
:2729 : NAD/TEST.MATCH.1
:-----
:2730 :
:2731 : 1F6:
:2732 : TEST.MATCH.1:
:2733 :
:2734 : BEN2/MISMATCH.L, ;IS MATCH ASSERTED?
:2735 : NAD/BEN2.1
:-----
:2736 :
:2737 : 1A0:
:2738 : =0**
:2739 : BEN2.1:
:2740 :
:2741 : INCR.DAR/ON, ;INCR DAR
:2742 : NAD/XFER.REQ ;FLAG THE CPU
:-----
:2743 :
:2744 : 1A4:
```

U 18  
U 18  
U 18  
  
U 18  
U 18  
U 18  
  
U 18  
  
U 18  
U 18  
U 18

```

:28745 :=1**
:28746 :BEN2.2:
:28747 :
:28748 : INCR.DAR/ON, ;INCR DAR IF BEN2 CONDITION ASSERTED
:28749 : NAD/BEN2.1 ;INCR DAR AGAIN IF BEN2 COND ASSERTED
:28750 :-----
:28751 :1D8:
:28752 :XFER.REQ:
:28753 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:28754 : NAD/DIAG.WAIT
:28755 :-----
:28756 :1D7:
:28757 :DIAG.WAIT:
:28758 :
:28759 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:28760 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:28761 :-----
:28762 :159:
:28763 :=*0*
:28764 :DIAG.WAIT1:
:28765 :
:28766 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:28767 : ; COMMAND FROM THE CPU
:28768 :-----
:28769 :15B:
:28770 :=*1*
:28771 :DIAG.WAIT2:
:28772 :
:28773 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:28774 : ; WAIT SOME MORE
:28775 :-----
:28776 :WAIT.IDC.T32.2:
:28777 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ FROM IDC
:28778 : JMP [IDC.DONE.T32.2] ;XFER REQ SET
:28779 : JMP [WAIT.IDC.T32.2] ;WAIT SOME MORE
:28780 :
:28781 :IDC.DONE.T32.2:
:28782 : MOV LS[SETCSR.F0] TO WR[0] ;CLEAR CSR F<2:0>
:28783 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:28784 : MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
:28785 :
:28786 :READ.DAR.T32.1:
:28787 : MISC [SET.PORT.1.0] ;SELECT THE IDC
:28788 : PORT.READ PORT.ADRS[DAR] ;SET UP TO READ THE DAR
:28789 : MOV PORT TO LS[PORT0] ;READ THE DATA FROM DAR AND
:28790 : ; SAVE IT IN LS
:28791 : MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
:28792 : MOV LS[T8] TO WR[1] ;SET UP EXPECTED DATA
:28793 :
:28794 : MOV WR[3] TO LS[ADDRESS.DATA] ;PRINT OTHER DATA
:28795 :
:28796 : JSR [CHECK.RESULT] ;CHECK THE DATA
:28797 : JMP [LOOP.T32.1] ;LOOP HERE IF ERR & LOE IS SET
:28798 :
:28799 : MOV LS[T9] TO WR[0] ;GET THE LOOP CNTR

```

U 17C6, DB00,1A  
 U 17C7, 897C,94  
 U 17C8, 897C,64

U 17C9, B700,15  
 U 17CA, 17A0,15  
 U 17CB, 90FA,15

U 17CC, 9090,15  
 U 17CD, 1784,15  
 U 17CE, 3B32,15

U 17CF, 3732,15  
 U 17D0, 3610,95

U 17D1, BE89,95

U 17D2, 0869,3C  
 U 17D3, 897A,A4

U 17D4, 3612,15

C 14  
Fiche 3 Frame C14  
Sequence 583  
Page 577

```

ZZ-ENKCF-1.4 : ENKCF.MIC Test 32 WRITE CHECK TEST (M8388 IDC MODULE)
: ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40
: ENKCF.MIC Test 32 WRITE CHECK TEST (M8388 IDC MODULE)

U 17D5, 2000,35 :28800 TST WR[0],
U 17D6, 897E,39 :28801 DT(LONG)&SET.ALU.CC
:28802 JMP.IF[EQL] TO [CLEANUP.T32] ;JUMP IF DONE
:28803
:28804 CMP LS[#1] WITH WR[0], ;LAST PAT?
U 17D7, 4E40,35 :28805 DT(LONG)&SET.ALU.CC
U 17D8, 897D,E1 :28806 JMP.IF[NEQ] TO [CHK.NXT.T32] ;JMP IF NOT LAST
:28807
:28808 ;SET UP THE LAST DATA PATTERN
U 17D9, 3651,95 :28809 MOV LS[#100] TO WR[3] ;SET UP THE DATA PAT
U 17DA, B640,15 :28810 MOV LS[#1] TO WR[0] ;SET UP THE EXPECTED DATA
U 17DB, 3E10,15 :28811 MOV WR[0] TO LS[T8] ;SAV EXP DATA IN LS[T8]
U 17DC, FC12,15 :28812 DEC LS[T9] ;DECREMENT THE LOOP CNTR
U 17DD, 897A,A4 :28813 JMP [TST.NXT.PAT.T32] ;GO TEST THE PATTERN
:28814
:28815 CHK.NXT.T32:
:28816 ;SET UP THE SECOND DATA PATTERN
U 17DE, B641,95 :28817 MOV LS[#1] TO WR[3] ;SET UP THE DATA PAT
U 17DF, B640,15 :28818 MOV LS[#1] TO WR[0] ;SET UP THE EXPECTED DATA
U 17E0, 3E10,15 :28819 MOV WR[0] TO LS[T8] ;SAV EXP IN LS[T8]
U 17E1, FC12,15 :28820 DEC LS[T9] ;DECREMENT THE LOOP CNTR
U 17E2, 897A,A4 :28821 JMP [TST.NXT.PAT.T32] ;GO TEST THE PATTERN
:28822
:28823 CLEANUP.T32:
U 17E3, 370E,15 :28824 MOV LS[SETCSR.F7] TO WR[0]
U 17E4, 17A0,15 :28825 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:28826 ;MAINT=0,CRDY=0,F<2:0>=111
:28827 ;CONTINUE TO BRANCH
U 17E5, 379E,15 :28828 MOV LS[SETCSR.R80] TO WR[0] ;SET UP THE CSR TO CONTINUE TO
U 17E6, 3700,95 :28829 MOV LS[SETCSR.F0] TO WR[1] ; BRANCH TO THE CORRECT ROUTINE
U 17E7, AEC2,15 :28830 BIS WR[1] TO WR[0]
U 17E8, 17A0,15 :28831 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:28832 ;F<2:0> = 000
:28833 ;UCLR MATCH
:28834 -----
:28835 -----
:28836 *****
:28837 * IDC *
:28838 *****
:28839 -----
:28840 ;018:
:28841 ;=000
:28842 ;DIAG.IDLE:
:28843
:28844 BEN0/F0,
:28845 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:28846 BEN2/F2,
:28847 NAD/DIAG.IDLE
:28848 -----
:28849 ;01F:
:28850 ;=111
:28851
:28852 BEN0/CRDY.L,
:28853 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:28854 NAD/TEST.FUNC7 ; DEPENDENT ON CSR <CRDY> AND <MAINT>

```



MICRO2 1M(01)

15-MAR-83 11:46:22

## ENKCF Micro-diagnostic for IDC module

Rev 01.40

Page 578

Test 32 WRITE CHECK TEST (M8388 IDC MODULE)

```

28855 :-----
28856 :079:
28857 : =0*1
28858 :
28859 : NAD/GROUP1
28860 :-----
28861 :182:
28862 : =0**
28863 :GROUP1:
28864 :
28865 : BEN2/R80.FORMAT.H, ;WHEN THE R80 FORMAT BIT SETS..BRANCH
28866 : NAD/GROUP1 ; TO NEXT WORD AND THEN BRANCH ON THE
28867 : ; FUNCTION BITS
28868 :-----
28869 :186:
28870 : =1**
28871 :
28872 : BEN0/F0,
28873 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
28874 : BEN2/F2,
28875 : NAD/GROUP1.F0
28876 :-----
28877 :028:
28878 : =000
28879 :GROUP1.F0:
28880 :
28881 :CLR.MATCH.1:
28882 :
28883 : UCON/CLR.MATCH, ;CLR THE MATCH CONDITION
28884 : NAD/TEST.MATCH.1
28885 :-----
28886 :1F6:
28887 :TEST.MATCH.1:
28888 :
28889 : BEN2/MISMATCH.L, ;IS MATCH ASSERTED?
28890 : NAD/BEN2.1
28891 :-----
28892 :1A0:
28893 : =0**
28894 :BEN2.1:
28895 :
28896 : INCR.DAR/ON, ;INCR DAR
28897 : NAD/XFER.REQ ;FLAG THE CPU
28898 :-----
28899 :1A4:
28900 : =1**
28901 :BEN2.2:
28902 :
28903 : INCR.DAR/ON, ;INCR DAR IF BEN2 CONDITION ASSERTED
28904 : NAD/BEN2.1 ;INCR DAR AGAIN IF BEN2 COND ASSERTED
28905 :-----
28906 :1D8:
28907 :XFER.REQ:
28908 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
28909 : NAD/DIAG.WAIT

```

U U U U U U U U U

ZZ-ENKCF-1.4  
: ENKCF.MCR  
: ENKCF.MIC

ENKCF.MIC

MICRO2 1M(01)

Test 32 WRITE CHECK TEST (M8388 IDC MODULE)

E 14

15-MAR-83 11:46:22

Fiche 3 Frame E14

Sequence 585

Rev 01.40

Page 579

ZZ-  
:  
:

```
:28910 :-----
:28911 :1D7:
:28912 :DIAG.WAIT:
:28913 :
:28914 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:28915 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:28916 :-----
:28917 :159:
:28918 :=*0*
:28919 :DIAG.WAIT1
:28920 :
:28921 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:28922 : ; COMMAND FROM THE CPU
:28923 :-----
:28924 :158:
:28925 :=*1*
:28926 :DIAG.WAIT2:
:28927 :
:28928 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:28929 : ; WAIT SOME MORE
:28930 :-----
:28931 :WAIT.IDC.T32.3:
:28932 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ FROM IDC
:28933 : JMP [IDC.DONE.T32.3] ;XFER REQ SET
:28934 : JMP [WAIT.IDC.T32.3] ;WAIT SOME MORE
:28935 :
:28936 :IDC.DONE.T32.3:
:28937 : MOV LS[SETCSR.F0] TO WR[0] ;CLEAR CSR F<2:0>
:28938 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:28939 : MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
:28940 :
:28941 : JSR [CLEANUP] ;CLEANUP CONDITIONS SET BY THIS
:28942 : ; TEST
:28943 :END.T32: ;END TEST
:28944
```

U 17E9, DB00,1A  
U 17EA, 097E,C4  
U 17EB, 097E,94

U 17EC, B700,15  
U 17ED, 17A0,15  
U 17EE, 90FA,15

U 17EF, 8875,3C

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

:28945 :PAGE "Test 33 ATTENTION REGISTER TEST 1 (M8388 IDC MODULE)"  
:28946 :++

:28947 :  
:28948 : Test Description:  
:28949 :

:28950 : This test is designed to test the Attention Register PAL. The  
:28951 : Attention bits <3:0> are CSR bits <19:16>. These bits can be set  
:28952 : only by the IDC microcode and are cleared by INIT or writing ones to  
:28953 : the CSR.  
:28954 : The Attention bits <3:0> will first be cleared by issuing a CLEAR IDC  
:28955 : to init the IDC. The CSR will then be read to insure that bits <19:16>  
:28956 : cleared.  
:28957 : Then, each attention bit beginning with <0>, will be set independently,  
:28958 : and all attention bits checked to verify that they are in the correct  
:28959 : state. After all attention bits are set, bits <19:16> will be written  
:28960 : as zeros. The CSR will then be read to insure that the ATTENTION BITS  
:28961 : remained set since the ATTN bits are write one to clear bits.

:28962 :  
:28963 : Assumptions:

:28964 :  
:28965 : It is assumed that all previous tests have run successfully.  
:28966 :

:28967 : Test Steps:

- :28968 :  
:28969 : 1. Set up error mask, error number and module number in LS  
:28970 : (for error printouts) and clear the previous error number in LS.  
:28971 : 2. INIT the IDC and force the IDC microcode to DIAG.IDLE.  
:28972 : 3. Check CSR bits <19:16> for zeros.  
:28973 : 4. Increment the error number.  
:28974 : 5. Select Drive 0.  
:28975 : 6. Set up the CSR to branch to the correct IDC microcode routine.

:28976 :  
:28977 :  
:28978 : \*\*\*\*\*  
:28979 : \* IDC \*  
:28980 : \*\*\*\*\*

:28981 :  
:28982 : IDC1. Execute an instruction with FUNC <35:32> = 0B,  
:28983 : USET.ATTN.  
:28984 : IDC2. Issue XFER.REQ to flag the CPU.

- :28985 :  
:28986 : 7. Wait for an XFER.REQ from IDC.  
:28987 : 8. Clear CSR F<2:0> and issue XFER GRANT to clear the XFER.REQ.  
:28988 : 9. Check CSR <19:16>.  
:28989 : 10. Select the next drive and go to Step 6 until all drives have been  
:28990 : tested.  
:28991 : 11. Increment the error number.  
:28992 : 12. Write zeros to CSR <19:16>.  
:28993 : 13. Check CSR <19:16> remain set.  
:28994 : 14. Clear IDC.  
:28995 : 15. End test.

:28996 :  
:28997 : Errors:

:28998 :  
:28999 : Printout:

[illegible]

```

29055 : with F<3:0> = 1011. These bits will be decoded and will assert
29056 : USET ATTN L. The inputs to the ATTN REG PAL will then be:
29057 : DRIVE SEL 1 H = H
29058 : DRIVE SEL 0 H = L
29059 : USET ATTN L = L
29060 : At the next P2 CLOCK L, BUS IO 18 H will assert. BUS IO 17 and
29061 : BUS IO 16 should still be asserted.
29062 : The CPU will then issue a READ.PORT PORT.ADRS[CSR] which is decoded
29063 : by the PORT INTERFACE REG PAL and DECODER and will assert SEL CSR L.
29064 : SEL CSR L is the enable for the ATTN REG PAL and allows BUS IO <19:16>
29065 : to be read as CSR <19:16>.
29066 :
29067 : ERROR 5 -
29068 : Drive 3 is selected by CSR <9:8>. The IDC will branch to an instruction
29069 : with F<3:0> = 1011. These bits will be decoded and will assert
29070 : USET ATTN L. The inputs to the ATTN REG PAL will then be:
29071 : DRIVE SEL 1 H = H
29072 : DRIVE SEL 0 H = H
29073 : USET ATTN L = L
29074 : At the next P2 CLOCK L, BUS IO 19 H will assert. BUS IO 18 and
29075 : BUS IO 17 and BSU IO 16 should still be asserted.
29076 : The CPU will then issue a READ.PORT PORT.ADRS[CSR] which is decoded
29077 : by the PORT INTERFACE REG PAL and DECODER and will assert SEL CSR L.
29078 : SEL CSR L is the enable for the ATTN REG PAL and allows BUS IO <19:16>
29079 : to be read as CSR <19:16>.
29080 :
29081 : ERROR 6-
29082 : ATTN BITS <3:0> are all asserted. The CPU will then issue a
29083 : PORT.WRITE PORT.ADRS[CSR] with bits <19:16> = 0000. This instruction
29084 : will be decoded by the PORT RD/WR DECODE PAL and will assert
29085 : WRITE CSR L. The inputs to the ATTN REG PAL will now be:
29086 : WRITE CSR L = L
29087 : BUS IO 19 H = L
29088 : BUS IO 18 H = L
29089 : BUS IO 17 H = L
29090 : BUS IO 16 H = L
29091 : This combination of inputs should not affect the latched up outputs of
29092 : the ATTN REG PAL and /TTN BIT <3:0> should all remain asserted.
29093 : --
29094 : *****
29095 : T.33:
29096 : MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT 15 IN WR[0] FOR
29097 : MOV WR[0] TO LS[CONTROL.STATUS] ; CONTROL AND STATUS WORD
29098 : ;SET BIT 15 IN CONTROL AND
29099 : ; STATUS WORD - BIT 15
29100 : ; INDICATES START OF TEST TO
29101 : ; CONSOLE
29102 : MISC [SET.CP.ATTN] ;ISSUE CPU ATTN TO CONSOLE
29103 : WAIT.T33.0:
29104 : JMP [WAIT.T33.0] ;LOOP HERE WAITING FOR
29105 : ; CONSOLE TO RESPOND
29106 :
29107 : CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
29108 : ; IN LS
29109 :

```

```

ZZ-ENKCF-1.4 : ENKCF.MIC Test 33 ATTENTION 15-MAR-83 I 14 fiche 3 Frame I14 Sequence 589
: ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40 Page 583
: ENKCF.MIC Test 33 ATTENTION REGISTER TEST 1 (M8388 IDC MODULE)

U 17F5, B64A,15 :29110 MOV LS[IDC] TO WR[0] ;STORE MODULE CODE IN LS TO
U 17F6, 3E8C,15 :29111 MOV WR[0] TO LS[MODULE.NUM] ; INDICATE IDC MODULE
:29112
U 17F7, B640,15 :29113 MOV LS[#1] TO WR[0] ;SET WRO = 1
U 17F8, BE82,15 :29114 MOV WR[0] TO LS[ERROR.NUMBER] ;SET UP ERROR NUMBER = 1 IN LS
:29115
:29116
U 17F9, 37BC,15 :29117 MOV LS[#FFFF0000] TO WR[0] ;SET UP THE MASK TO CHECK
U 17FA, 3E8A,15 :29118 MOV WR[0] TO LS[ERROR.MASK] ; BIT <19:16>
:29119
U 17FB, B670,15 :29120 MOV LS[OTHER.DATA] TO WR[0] ;SET UP TO PRINT OTHER DATA
U 17FC, 3E80,15 :29121 MOV WR[0] TO LS[CONTROL.STATUS] ; IF ERROR
:29122
:29123
U 17FD, 8872,EC :29124 LOOP.T33.1:
:29125 JSR [DIAG.CODE] ;FORCE THE IDC TO DIAG.IDLE
:29126 ;INIT SHOULD CLEAR CSR<19:16>
:29127
:29128 READ.CSR.T33.1:
U 17FE, 9090,15 :29129 MISC [SET.PORT.1.0] ;SELECT THE IDC
U 17FF, 9780,15 :29130 PORT.READ PORT.ADRS[CSR] ;SET UP TO READ THE CSR
U 1800, 3B32,15 :29131 MOV PORT TO LS[PORT0] ;READ THE DATA FROM CSR AND
:29132 ; SAVE IT IN LS
U 1801, 3732,15 :29133 MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
U 1802, 2F82,95 :29134 CLR WR[1] ;SET UP THE EXPECTED DATA
U 1803, 3E88,95 :29135 MOV WR[1] TO LS[ADDRESS.DATA] ;CLR OTHER DATA
U 1804, 0869,3C :29136 JSR [CHECK.RESULT] ;GO CHECK THE DATA
U 1805, 097F,D4 :29137 JMP [LOOP.T33.1] ;LOOP HERE IF ERR & LOE IS SET
:29138
:29139
:29140 LOOP1.T33.2:
U 1806, 3642,15 :29141 MOV LS[#2] TO WR[0] ;ERROR 2
U 1807, BE82,15 :29142 MOV WR[0] TO LS[ERROR.NUMBER]
:29143
U 1808, B714,15 :29144 MOV LS[SETCSR.DS0] TO WR[0] ;GET THE FIRST DRIVE NUM
U 1809, 3E10,15 :29145 MOV WR[0] TO LS[T8] ;SAVE IN LS[T8]
U 180A, 3660,15 :29146 MOV LS[#10000] TO WR[0] ;SET UP THE EXPECTED DATA
U 180B, BE12,15 :29147 MOV WR[0] TO LS[T9] ;SAV EXP DATA IN LS[T9]
:29148
:29149
U 180C, 370E,15 :29150 TST.NXT.T33:
U 180D, 17A0,15 :29151 MOV LS[SETCSR.F7] TO WR[0] ;SET UP THE CSR TO BRANCH TO
:29152 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ; THE CORRECT IDC ROUTINE
:29153
:29154 ;CONTINUE TO BRANCH
U 180E, 379E,15 :29155 MOV LS[SETCSR.R80] TO WR[0]
:29156
U 180F, 3712,95 :29157 MOV LS[SETCSR.CRDY] TO WR[1]
U 1810, AEC2,15 :29158 BIS WR[1] TO WR[0]
U 1811, B708,95 :29159 MOV LS[SETCSR.F4] TO WR[1]
U 1812, AEC2,15 :29160 BIS WR[1] TO WR[0]
U 1813, 3610,95 :29161 MOV LS[T8] TO WR[1] ;INCLUDE THE DRIVE SELECT
U 1814, AEC2,15 :29162 BIS WR[1] TO WR[0]
U 1815, 17A0,15 :29163 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:29164 ;F<2:0>=100
:29165 ;USET.ATTN....drive number selected by CSR <9:8>

```

```

29165 -----
29166
29167 *****
29168 * IDC *
29169 *****
29170 -----
29171 018:
29172 =000
29173 DIAG.IDLE:
29174
29175 BEN0/F0,
29176 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
29177 BEN2/F2,
29178 NAD/DIAG.IDLE
29179 -----
29180 01F:
29181 =111
29182
29183 BEN0/CRDY.L,
29184 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
29185 NAD/TEST.FUNC7 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
29186 -----
29187 079:
29188 =0*1
29189
29190 NAD/GROUP1
29191 -----
29192 182:
29193 =0**
29194 GROUP1:
29195
29196 BEN2/R80.FORMAT.H, ;WHEN THE R80 FORMAT BIT SETS..BRANCH
29197 NAD/GROUP1 ; TO NEXT WORD AND THEN BRANCH ON THE
29198 ; FUNCTION BITS
29199 -----
29200 186:
29201 =1**
29202
29203 BEN0/F0,
29204 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
29205 BEN2/F2,
29206 NAD/GROUP1.F0
29207 -----
29208 02C:
29209 =100
29210 USET.ATTN:
29211
29212 FUNC/SET.ATTN, ;SET ATTENTION BE SURE CRDY SET TO ALLOW
29213 ; INTERRUPTS
29214 NAD/XFER.REQ ;SEND XFER REQ TO FLAG THE CPU
29215 -----
29216 1D8:
29217 XFER.REQ:
29218 UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
29219 NAD/DIAG.WAIT

```

[illegible]

```

:29220 -----
:29221 :1D7:
:29222 :DIAG.WAIT:
:29223
:29224 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:29225 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:29226 -----
:29227 :159:
:29228 :=*0*
:29229 :DIAG.WAIT1-
:29230
:29231 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:29232 : ; COMMAND FROM THE CPU
:29233 -----
:29234 :15B:
:29235 :=*1*
:29236 :DIAG.WAIT2:
:29237
:29238 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:29239 : ; WAIT SOME MORE
:29240 -----
:29241 :WAIT.IDC.T33.1:
:29242 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ FROM IDC
:29243 : JMP [IDC.DONE.T33.1] ;XFER REQ SET
:29244 : JMP [WAIT.IDC.T33.1] ;WAIT SOME MORE
:29245
:29246 :IDC.DONE.T33.1:
:29247 : MOV LS[SETCSR.F0] TO WR[0] ;CLEAR CSR F<2:0>
:29248 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:29249 : MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
:29250
:29251 :READ.CSR.T33.2:
:29252 : MISC [SET.PORT.I.0] ;SELECT THE IDC
:29253 : PORT.READ PORT.ADRS[CSR] ;SET UP TO READ THE CSR
:29254 : MOV PORT TO LS[PORT0] ;READ THE DATA FROM CSR AND
:29255 : ; SAVE IT IN LS
:29256 : MOV LS[T8] TO WR[0] ;PUT DRIVE NUM IN OTHER DATA
:29257 : MOV WR[0] TO LS[ADDRESS.DATA] ;SET UP RECEIVED DATA
:29258 : MOV LS[PORT0] TO WR[0] ;SET UP THE EXPECTED DATA
:29259 : MOV LS[T9] TO WR[1] ;CHECK THE DATA
:29260 : JSR [CHECK.RESULT] ;LOOP HERE IF ERR & LOE IS SET
:29261 : JMP [LOOP.T33.2]
:29262
:29263 : INC LS[ERROR.NUMBER] ;NEXT ERROR NUMBER
:29264
:29265 : MOV LS[T8] TO WR[2] ;GET THE CURRENT DRIVE NUMBER
:29266 : ADD LS[#100] TO WR[2] ;SELECT THE NEXT DRIVE NUMBER
:29267 : CMP LS[#400] WITH WR[2], ;DONE?
:29268 : DT(LONG)&SET.ALU.CC
:29269 : JMP.IF[EQL] TO [CHK.CLR.T33] ;JUMP IF DONE
:29270
:29271 : MOV WR[2] TO LS[T8] ;SAV THE NEW DRIVE SELECT
:29272 : CMP LS[#100] WITH WR[2], ;DRIVE 1 SELECTED?
:29273 : DT(LONG)&SET.ALU.CC
:29274 : JMP.IF[NEQ] TO [NXT.EXP.T33.1] ;JMP IF NOT

```

U 1816, DB00,1A  
 U 1817, 0981,94  
 U 1818, 0981,64

U 1819, B700,15  
 U 181A, 17A0,15  
 U 181B, 90FA,15

U 181C, 9090,15  
 U 181D, 9780,15  
 U 181E, 3832,15

U 181F, B610,15  
 U 1820, BE88,15  
 U 1821, 3732,15  
 U 1822, B612,95  
 U 1823, 0869,3C  
 U 1824, 0984,64

U 1825, FF82,15

U 1826, 3611,15  
 U 1827, 4051,15

U 1828, CE55,35  
 U 1829, 8983,89

U 182A, BE11,15

U 182B, 4E51,35  
 U 182C, 8983,01



```

:29275
U 182D, B7BE,15 :29276 MOV LS[#30000] TO WR[0] ;SET UP THE EXPECTED DATA
U 182E, BE12,15 :29277 MOV WR[0] TO LS[T9] ;SAV EXP DATA IN LS[T9]
U 182F, 8980,C4 :29278 JMP [TST.NXT.T33] ;GO TEST NEXT PAT
:29279
:29280 NXT.EXP.T33.1:
:29281 CMP LS[#200] WITH WR[2], ;DRIVE 2 SELECTED?
U 1830, CE53,35 :29282 DT(LONG)&SET.ALU.CC
U 1831, 8983,51 :29283 JMP.[IF[NEQ] TO [NXT.EXP.T33.2] ;JMP IF NOT
:29284
U 1832, B7C0,15 :29285 MOV LS[#70000] TO WR[0] ;SET UP THE EXPECTED DATA
U 1833, BE12,15 :29286 MOV WR[0] TO LS[T9] ;SAV EXP DATA IN LS[T9]
U 1834, 8980,C4 :29287 JMP [TST.NXT.T33] ;GO TEST NXT PAT
:29288
:29289 NXT.EXP.T33.2:
U 1835, 37C2,15 :29290 MOV LS[#F0000] TO WR[0] ;SET THE EXP DAT FOR DRV 3
U 1836, BE12,15 :29291 MOV WR[0] TO LS[T9] ;SAV IN LS[T9]
U 1837, 8980,C4 :29292 JMP [TST.NXT.T33] ;GO TEST NEXT PAT
:29293
:29294 CHK.CLR.T33:
U 1838, 3792,15 :29295 MOV LS[#6] TO WR[0] ;ERROR 6
U 1839, BE82,15 :29296 MOV WR[0] TO LS[ERROR.NUMBER]
:29297
:29298
U 183A, 3678,15 :29299 MOV LS[BIT28] TO WR[0] ;BIT 28 PREVENTS IDC TIMEOUTS
U 183B, 17A0,15 :29300 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE ZEROS TO CSR<19:16>
:29301 ; AND CHECK THAT ATTN<3:0>
:29302 ; REMAIN SET
:29303 READ.CSR.T33.3:
U 183C, 9090,15 :29303 MISC [SET.PORT.I.0] ;SELECT THE IDC
U 183D, 9780,15 :29304 PORT.READ PORT.ADRS[CSR] ;SET UP TO READ THE CSR
U 183E, 3B32,15 :29305 MOV PORT TO LS[PORT0] ;READ THE DATA FROM CSR AND
:29306 ; SAVE IT IN LS
U 183F, 3732,15 :29307 MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
U 1840, B612,95 :29308 MOV LS[T9] TO WR[1] ;SET UP THE EXPECTED DATA
U 1841, 2F85,15 :29309 CLR WR[2]
U 1842, 3E89,15 :29310 MOV WR[2] TO LS[ADDRESS.DATA] ;CLR OTHER DATA
U 1843, 086,3C :29311 JSR [CHECK.RESULT] ;CHECK THE DATA
U 1844, 8984,84 :29312 JMP [LOOP.T33.6] ;LOOP HERE IF ERR & LOE IS SET
:29313
U 1845, 0986,24 :29314 JMP [CLEANUP.T33] ;CLEANUP AND END TEST
:29315
:29316
:29317 LOOP.T33.2:
U 1846, 8872,EC :29318 JSR [DIAG.CODE] ;INIT THE IDC TO CLR ATTN BITS
U 1847, 8980,64 :29319 JMP [LOOP1.T33.2] ;LOOP ON ERROR
:29320
:29321
:29322 LOOP.T33.6:
:29323 ;SET ALL ATTN BITS
U 1848, B715,95 :29324 MOV LS[SETCSR.DS0] TO WR[3]
:29325
:29326 SET.ATTN.T33:
:29327 ;SET UP THE CSR TO BRANCH TO THE CORRECT IDC ROUTINE
:29328
U 1849, 370E,15 :29329 MOV LS[SETCSR.F7] TO WR[0]

```

[illegible]

```

:29385 :USET.ATTN:
:29386 :
:29387 : FUNC/SET.ATTN, ;SET ATTENTION BE SURE CRDY SET TO ALLOW
:29388 : NAD/XFER.REQ ;INTERRUPTS
:29389 : ;SEND XFER REQ TO FLAG THE CPU
:29390 :-----
:29391 :1D8:
:29392 :XFER.REQ:
:29393 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:29394 : NAD/DIAG.WAIT
:29395 :-----
:29396 :1D7:
:29397 :DIAG.WAIT:
:29398 :
:29399 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:29400 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:29401 :-----
:29402 :159:
:29403 :=*0*
:29404 :DIAG.WAIT1:
:29405 :
:29406 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:29407 : ; COMMAND FROM THE CPU
:29408 :-----
:29409 :15B:
:29410 :=*1*
:29411 :DIAG.WAIT2:
:29412 :
:29413 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:29414 : ; WAIT SOME MORE
:29415 :-----
:29416 :WAIT.IDC.T33.2:
:29417 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ FROM IDC
:29418 : JMP [IDC.DONE.T33.2] ;XFER REQ SET
:29419 : JMP [WAIT.IDC.T33.2] ;WAIT SOME MORE
:29420 :
:29421 :IDC.DONE.T33.2:
:29422 : MOV LS[SETCSR.F0] TO WR[0] ;CLEAR CSR F<2:0>
:29423 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:29424 : MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
:29425 :
:29426 : ADD LS[#100] TO WR[3] ;SELECT THE NEXT DRIVE
:29427 : CMP LS[#400] WITH WR[3], ;ALL BITS BEEN SET?
:29428 : DT(LONG)&SET.ALU.CC
:29429 : JMP.IF[NEQ] TO [SET.ATTN.T33] ;JMP IF NO
:29430 :
:29431 : MOV LS[BIT28] TO WR[0] ;WRITE ZERO TO CSR <19:16>
:29432 : ; BIT 28 PREVENTS IDC TIMEOUTS
:29433 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0]
:29434 :
:29435 :READ.CSR.T33.4:
:29436 : MISC [SET.PORT.I.0] ;SELECT THE IDC
:29437 : PORT.READ PORT.ADRS[CSR] ;SET UP TO READ THE CSR
:29438 : MOV PORT TO LS[PORT0] ;READ THE DATA FROM CSR AND
:29439 : ; SAVE IT IN LS

```

U 1850, DB00,1A  
 U 1851, 8985,34  
 U 1852, 8985,04

U 1853, B700,15  
 U 1854, 17A0,15  
 U 1855, 90FA,15

U 1856, C051,95  
 U 1857, 4E55,85  
 U 1858, 0984,91

U 1859, 3678,15  
 U 185A, 17A0,15

U 185B, 9090,15  
 U 185C, 9780,15  
 U 185D, 3B32,15

ZZ-ENKCF-1.4 : ENKCF.MIC Test 33 ATTENTION 15-MAR-83 B 15 Fiche 3 Frame B15 Sequence 595  
: ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40 Page 589  
: ENKCF.MIC Test 33 ATTENTION REGISTER TEST 1 (M8388 IDC MODULE)

|                 |        |                         |                                |
|-----------------|--------|-------------------------|--------------------------------|
| U 185E, 3732,15 | :29440 | MOV LS[PORT0] TO WR[0]  | :SET UP RECEIVED DATA          |
| U 185F, B7C2,95 | :29441 | MOV LS[#F0000] TO WR[1] | :SET UP EXPECTED DATA          |
| U 1860, 0869,3C | :29442 | JSR [CHECK.RESULT]      | :CHECK THE DATA                |
| U 1861, 8984,84 | :29443 | JMP [LOOP.T33.6]        | :LOOP HERE ON ERR & LOE IS SET |
|                 | :29444 |                         |                                |
|                 | :29445 | CLEANUP.T33:            |                                |
| U 1862, 8875,3C | :29446 | JSR [CLEANUP]           | :CLEANUP THE CONDITIONS SET BY |
|                 | :29447 |                         | : THIS TEST                    |
|                 | :29448 | END.T33:                | :END TEST                      |
|                 | :29449 |                         |                                |
|                 | :29450 |                         |                                |
|                 | :29451 |                         |                                |

ZZ  
:  
:

CCC  
CCC  
CC  
CCCCC

:29452 :PAGE "Test 34 ATTENTION REGISTER TEST 2 (M8388 IDC MODULE)"

:29453 :++

:29454 :

:29455 :Test Description:

:29456 :

:29457 :This test is designed to test the Attention Register PAL. The  
:29458 :Attention bits <3:0> are CSR bits <19:16>. These bits will be set  
:29459 :by IDC microcode. After all attention bits are set, each will be  
:29460 :cleared one at a time beginning with attention bit <3>. All bits will  
:29461 :be checked each time to verify that they are in the correct state.  
:29462 :Then all bits will again be set by IDC microcode and an INIT of the  
:29463 :IDC will be performed. The Attention bits will be checked to insure  
:29464 :that they are all cleared by INIT.

:29465 :

:29466 :Assumptions:

:29467 :

:29468 :It is assumed ATTENTION REG TEST 1 has been run successfully.

:29469 :

:29470 :Test Steps:

:29471 :

- :29472 :1. Set up error mask, error number and module number in LS  
:29473 : (for error printouts) and clear the previous error number in LS.  
:29474 :2. Force the IDC microcode to DIAG.IDLE.  
:29475 :3. Select each drive and set the associated ATTENTION BITS.  
:29476 :4. Clear Attention bit <3>.  
:29477 :5. Check CSR <19:16>.  
:29478 :6. Clear the next Attention bit.  
:29479 :7. Check CSR <19:16>.  
:29480 :8. Execute Steps 6 and 7 until all Attention bits have been cleared.  
:29481 :9. Increment the error number.  
:29482 :10. Select each drive and the associated ATTENTION BITS.  
:29483 :11. Clear IDC.  
:29484 :12. Check CSR <19:16> = 0000.  
:29485 :13. Clear IDC.  
:29486 :14. End test.

:29487 :

:29488 :Errors:

:29489 :

:29490 :

:29491 :

:29492 :

:29493 :

:29494 :

:29495 :

:29496 :

:29497 :

:29498 :

:29499 :

:29500 :

:29501 :

:29502 :

:29503 :

:29504 :

:29505 :

:29506 :

ERROR 1 -

ATTN BITS <3:0> are all set. The CPU will then issue a  
PORT.WRITE PORT.ADRS[CSR] instruction with BIT <19> = 1.  
This instruction will be decoded by the PORT RD/WR DECODE PAL  
and will assert WRITE CSR L. The inputs to the ATTN REG PAL will be:  
WRITE CSR L = L

Printout:

EXPECTED  
CSR

RECEIVED  
CSR

OTHER DATA  
DRV NUM SELECTED  
WR[2]

ERROR 1 - The Attention bits did not clear independently.

ERROR 2 - INIT did not clear the ATTENTION BITS.

Debug:

```

:29507 : BUS IO 19 H = H
:29508 : BUS IO 18 H = L
:29509 : BUS IO 17 H = L
:29510 : BUS IO 16 H = L
:29511 : At the next P2 CLOCK L, ATTN BIT <3> should be cleared and <2:0> should
:29512 : remain set. The CPU will then issue a PORT.READ PORT.ADRS[CSR] which
:29513 : will be decoded by the PORT INTERFACE REG PAL and DECODER and will
:29514 : assert SEL CSR L. SEL CSR L is the enable for the ATTN REG PAL and
:29515 : allows BUS IO <19:16> to be read as CSR <19:16>.
:29516 :
:29517 : ATTN BIT <2> is then cleared by writing a one to CSR <18>.
:29518 : ATTN BIT <3> should remain clear and ATTN <1:0> should remain set.
:29519 :
:29520 : ATTN BIT <1> is then cleared by writing a one to CSR <17>.
:29521 : ATTN BITS <3:2> should remain clear and ATTN BIT <0> should remain set.
:29522 :
:29523 : ATTN BIT <0> is then cleared by writing a one to CSR <16>.
:29524 : ATTN BITS <3:1> should remain clear.
:29525 :
:29526 : ERROR 2 -
:29527 : ATTN BITS <3:0> are all set.
:29528 : The CPU issues a series of PORT.CONTROL [CLEAR.IDC] instructions which
:29529 : are decoded by the PORT RD/WR DECODE PAL. This will assert CLEAR IDC L.
:29530 : At the next P2 CLOCK L, the INIT FF will assert INIT SYNC 1 H. At the
:29531 : next clock INIT H is asserted. INIT H is an input to the ATTN REG PAL
:29532 : and when it asserts ATTN BITS <3:0> will be cleared.
:29533 : --
:29534 : *****
:29535 : T.34:
:29536 : MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT 15 IN WR[0] FOR
:29537 : MOV WR[0] TO LS[CONTROL.STATUS] ;CONTROL AND STATUS WORD
:29538 : ;SET BIT 15 IN CONTROL AND
:29539 : ;STATUS WORD - BIT 15
:29540 : ;INDICATES START OF TEST TO
:29541 : ;CONSOLE
:29542 : MISC [SET.CP.ATTN] ;ISSUE CPU ATTN TO CONSOLE
:29543 : WAIT.T34.0:
:29544 : JMP [WAIT.T34.0] ;LOOP HERE WAITING FOR
:29545 : ;CONSOLE TO RESPOND
:29546 :
:29547 : CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
:29548 : ;IN LS
:29549 :
:29550 : MOV LS[IDC] TO WR[0] ;STORE MODULE CODE IN LS TO
:29551 : MOV WR[0] TO LS[MODULE.NUM] ;INDICATE IDC MODULE
:29552 :
:29553 :
:29554 : MOV LS[#1] TO WR[0] ;SET WRO = 1
:29555 : MOV WR[0] TO LS[ERROR.NUMBER] ;SET UP ERROR NUMBER = 1 IN LS
:29556 :
:29557 : MOV LS[#FFFF0000] TO WR[0] ;GET THE MASK TO CHECK
:29558 : MOV WR[0] TO LS[ERROR.MASK] ;BITS <19:16>
:29559 :
:29560 : MOV LS[OTHER.DATA] TO WR[0] ;SET UP TO PRINT OTHER DATA
:29561 : MOV WR[0] TO LS[CONTROL.STATUS] ;IF ERROR

```

```

:29562
:29563
U 1870, 8872,EC :29564 JSR [DIAG.CODE] ;FORCE THE IDC TO DIAG.IDLE
:29565
:29566 LOOP.T34.1:
:29567 ;SET ALL ATTN BITS
:29568
U 1871, B715,95 :29569 MOV LS[SETCSR.DS0] TO WR[3] ;SELECT DRIVE 0
:29570
:29571 SET.ATTN.T34:
:29572
U 1872, 370E,15 :29573 MOV LS[SETCSR.F7] TO WR[0]
U 1873, 17A0,15 :29574 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:29575 ;MAINT=0,CRDY=0,F<2:0>=111
:29576 ;CONTINUE TO BRANCH
:29577 MOV LS[SETCSR.R80] TO WR[0] ;SET UP THE CSR TO CONTINUE TO
U 1874, 379E,15 :29578 MOV LS[SETCSR.F4] TO WR[1] ; BRANCH TO THE CORRECT ROUTINE
U 1875, B708,95 :29579 BIS WR[1] TO WR[0]
U 1876, AEC2,15 :29580 BIS WR[3] TO WR[0] ;INCLUDE THE DRIVE SELECT
U 1877, 2EC6,15 :29581 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
U 1878, 17A0,15 :29582 ;F<2:0> = 100
:29583 ;USET ATTN...DRIVE SELECTED BY CSR <9:8>
:29584 -----
:29585 -----
:29586 *****
:29587 * IDC *
:29588 *****
:29589 -----
:29590 :018:
:29591 =000
:29592 DIAG.IDLE:
:29593
:29594 BEN0/F0,
:29595 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:29596 BEN2/F2,
:29597 NAD/DIAG.IDLE
:29598 -----
:29599 :01F:
:29600 =111
:29601
:29602 BEN0/CRDY.L,
:29603 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:29604 NAD/TEST.FUNC7 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:29605 -----
:29606 :079:
:29607 =0*1
:29608
:29609 NAD/GROUP1
:29610 -----
:29611 :182:
:29612 =0**
:29613 GROUP1:
:29614
:29615 BEN2/R80.FORMAT.H, ;WHEN THE R80 FORMAT BIT SETS..BRANCH
:29616 NAD/GROUP1 ; TO NEXT WORD AND THEN BRANCH ON THE

```

```

29617 : ; FUNCTION BITS
29618 : -----
29619 : 186:
29620 : =1**
29621 :
29622 : BEN0/F0,
29623 : BEN1/F1, ; BRANCH ON THE CSR F<2:0> BITS
29624 : BEN2/F2,
29625 : NAD/GROUP1.F0
29626 : -----
29627 : 02C:
29628 : =100
29629 : USET.ATTN:
29630 :
29631 : FUNC/SET.ATTN, ; SET ATTENTION BE SURE CRDY SET TO ALLOW
29632 : ; INTERRUPTS
29633 : NAD/XFER.REQ ; SEND XFER REQ TO FLAG THE CPU
29634 : -----
29635 : 1D8:
29636 : XFER.REQ:
29637 : UCMD/SET.XFER.RQST, ; SEND XFER REQ TO THE CPU
29638 : NAD/DIAG.WAIT
29639 : -----
29640 : 1D7:
29641 : DIAG.WAIT:
29642 :
29643 : BEN1/XFER.REQ, ; WAIT FOR THE CPU TO ISSUE XFER GRANT
29644 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
29645 : -----
29646 : 159:
29647 : =*0*
29648 : DIAG.WAIT1:
29649 :
29650 : NAD/DIAG.IDLE ; RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
29651 : ; COMMAND FROM THE CPU
29652 : -----
29653 : 15B:
29654 : =*1*
29655 : DIAG.WAIT2:
29656 :
29657 : NAD/DIAG.WAIT ; XFER GRANT HAS NOT BEEN ISSUED....
29658 : ; WAIT SOME MORE
29659 : -----
29660 : WAIT.IDC.T34.1:
29661 : SKIP.IF[NO.FAST.INTERRUPT] ; WAIT FOR XFER REQ FROM IDC
29662 : JMP [IDC.DONE.T34.1] ; XFER REQ SET
29663 : JMP [WAIT.IDC.T34.1] ; WAIT SOME MORE
29664 :
29665 : IDC.DONE.T34.1:
29666 : MOV LS[SETCSR.F0] TO WR[0] ; CLEAR CSR F<2:0>
29667 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ; WRITE THE CSR
29668 : MISC2 [TRANSFER.GRANT] ; SEND GRANT TO CLR THE XFER REQ
29669 :
29670 :
29671 : ADD LS[#100] TO WR[3] ; SELECT THE NEXT DRIVE

```

U 1879, DB00,1A  
 U 187A, 0987,C4  
 U 187B, 0987,94

U 187C, B700,15  
 U 187D, 17A0,15  
 U 187E, 90FA,15

U 187F, C051,95



**7777777777**

```

ZZ-ENKCF-1.4 : ENKCF.MIC Test 34 ATTENTION 15-MAR-83 H 15
: ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module
: ENKCF.MIC Test 34 ATTENTION REGISTER TEST 2 (M8388 IDC MODULE)
Sequence 601 Page 595

U 189E, 898A,51 :29727 JMP.IF[NEQ] TO [NXT.BIT.T34.2] ;JMP IF NOT
:29728
U 189F, B662,15 :29729 MOV LS[#20000] TO WR[0] ;SET UP THE CSR WRITE DATA
U 18A0, 3E10,15 :29730 MOV WR[0] TO LS[T8] ;SAVE IT IN LS[T8]
:29731
U 18A1, 3660,15 :29732 MOV LS[#10000] TO WR[0] ;SET UP THE EXPECTED DATA
U 18A2, BE12,15 :29733 MOV WR[0] TO LS[T9] ;SAVE THE EXP DATA IN LS[T9]
:29734
U 18A3, 2105,15 :29735 DEC WR[2] ;DECREMENT THE LOOP CNTR
:29736
U 18A4, 8988,74 :29737 JMP [TST.NXT.T34] ;GO TEST NEXT BIT
:29738
:29739 NXT.BIT.T34.2:
U 18A5, 3660,15 :29740 MOV LS[#10000] TO WR[0] ;SET UP THE CSR WRT DATA TO
U 18A6, 3E10,15 :29741 MOV WR[0] TO LS[T8] ;TEST ATTN BIT <0>
:29742
U 18A7, 6512,15 :29743 CLR LS[T9] ;SET UP THE EXPECTED DATA
:29744
U 18A8, 2105,15 :29745 DEC WR[2] ;DECREMENT THE LOOP CNTR
U 18A9, 8988,74 :29746 JMP [TST.NXT.T34] ;GO TEST THE NEXT BIT
:29747
:29748 TST.INIT.T34:
:29749 ;SET ALL ATTN BITS
:29750
U 18AA, FF82,15 :29751 INC LS[ERROR.NUMBER] ;SET UP FOR ERROR 2
U 18AB, B715,95 :29752 MOV LS[SETCSR.DS0] TO WR[3]
:29753
:29754 SET.ATTN.T34.2:
:29755
U 18AC, 370E,15 :29756 MOV LS[SETCSR.F7] TO WR[0]
U 18AD, 17A0,15 :29757 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:29758 ;MAINT=0,CRDY=0,F<2:0>=111
:29759 ;CONTINUE TO BRANCH
U 18AE, 379E,15 :29760 MOV LS[SETCSR.R80] TO WR[0] ;SET UP THE CSR TO CONTINUE TO
U 18AF, B708,95 :29761 MOV LS[SETCSR.F4] TO WR[1] ;BRANCH TO THE CORRECT ROUTINE
U 18B0, AEC2,15 :29762 BIS WR[1] TO WR[0]
U 18B1, 2EC6,15 :29763 BIS WR[3] TO WR[0] ;INCLUDE THE DRIVE SELECT
U 18B2, 17A0,15 :29764 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:29765 ;F<2:0> = 100
:29766 ;USET ATTN...DRIVE SELECTED BY CSR <9:8>
:29767 -----
:29768 -----
:29769 *****
:29770 * IDC *
:29771 *****
:29772 -----
:29773 :018:
:29774 :=000
:29775 :DIAG.IDLE:
:29776
:29777 :BEN0/F0,
:29778 :BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:29779 :BEN2/F2,
:29780 :NAD/DIAG.IDLE
:29781 -----

```

U  
U  
U  
U  
U

cccccc

```

:29880 .PAGE 'Test 35 CSR OPERATION INCOMPLETE (OPI) BITS TEST (M8388 IDC MODULE)'
:29881 .++
:29882
:29883 .Test Description:
:29884
:29885 .This test will verify that CSR bits <10>, <12> and <15>
:29886 .will set when the IDC issues a SET.OPI. It will also verify that
:29887 .CSR bit <10> clears when CSR <CRDY> is cleared or when INIT is
:29888 .issued.
:29889
:29890 .Assumptions:
:29891
:29892 .It is assumed that all previous tests have run successfully.
:29893
:29894 .Test Steps:
:29895
:29896 .1. Set up error mask, error number and module number in LS
:29897 .(for error printouts) and clear the previous error number in LS.
:29898 .2. Force the IDC microcode to DIAG.IDLE.
:29899 .3. Set up the CSR to branch to the correct IDC microcode routine that
:29900 .will clear match.
:29901
:29902
:29903 .*****
:29904 .* IDC *
:29905 .*****
:29906 .IDC1. UCLR MATCH.
:29907 .IDC2. Send XFER REQ to flag the CPU.
:29908
:29909 .4. Wait for XFER REQ from IDC.
:29910 .5. Clear CSR F<2:0> and send XFER GRANT to clear XFER.REQ.
:29911 .6. Set up the CSR to branch to the IDC routine that will SET OPI.
:29912
:29913 .*****
:29914 .* IDC *
:29915 .*****
:29916
:29917 .IDC3. Execute an instruction with UCM <2:0> = 2, SET.OPI.
:29918 .IDC4. Issue XFER.REQ to flag the CPU.
:29919
:29920 .7. Wait for a XFER.REQ from IDC.
:29921 .8. Clear CSR F<2:0> and send XFER GRANT to clear XFER.REQ.
:29922 .9. Check CSR bits <15> <12> <10>.
:29923 .10. Increment the error number.
:29924 .11. Clear CSR <CRDY>.
:29925 .12. Check CSR bits <10>.
:29926 .13. Increment the error number.
:29927 .14. Set OPI.
:29928 .15. INIT the IDC.
:29929 .16. Check CSR bit <10>.
:29930 .17. Clear IDC.
:29931 .18. End test.
:29932
:29933 .Errors:
:29934

```

[illegible]

```

U 18CC, 65A0,15 :29990 CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
 :29991 ; IN LS
 :29992
U 18CD, B64A,15 :29993 MOV LS[IDC] TO WR[0] ;STORE MODULE CODE IN LS TO
U 18CE, 3E8C,15 :29994 MOV WR[0] TO LS[MODULE.NUM] ; INDICATE IDC MODULEF
 :29995
 :29996
U 18CF, B640,15 :29997 MOV LS[#1] TO WR[0] ;SET WRO = 1
U 18D0, BE82,15 :29998 MOV WR[0] TO LS[ERROR.NUMBER] ;SET UP ERROR NUMBER = 1 IN LS
 :29999
U 18D1, B7C6,15 :30000 MOV LS[#FFFF68FF] TO WR[0] ;SET UP THE MASK TO CHECK
U 18D2, 3E8A,15 :30001 MOV WR[0] TO LS[ERROR.MASK] ; BITS <15> <12> <10>
 :30002
U 18D3, 8872,EC :30003 JSR [DIAG.CODE] ;FORCE THE IDC TO DIAG.IDLE
 :30004 ; AND INIT THE IDC
 :30005
U 18D4, 370E,15 :30006 MOV LS[SETCSR.F7] TO WR[0] ;BRANCH TO THE IDC ROUTINE THAT
U 18D5, 17A0,15 :30007 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ; WILL CLEAR A MATCH CONDITION
 :30008
 :30009 ;CONTINUE TO BRANCH
 :30010
U 18D6, 379E,15 :30011 MOV LS[SETCSR.R80] TO WR[0]
U 18D7, 3700,95 :30012 MOV LS[SETCSR.F0] TO WR[1]
U 18D8, AEC2,15 :30013 BIS WR[1] TO WR[0]
U 18D9, 17A0,15 :30014 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
 :30015 ;F<2:0> = 000
 :30016 ;UCLR MATCH
 :30017 -----
 :30018 -----
 :30019 *****
 :30020 * IDC *
 :30021 *****
 :30022 -----
 :30023 ;018:
 :30024 ;=000
 :30025 ;DIAG.IDLE:
 :30026
 :30027 BEN0/F0,
 :30028 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
 :30029 BEN2/F2,
 :30030 NAD/DIAG.IDLE
 :30031 -----
 :30032 ;01F:
 :30033 ;=111
 :30034
 :30035 BEN0/CRDY.L,
 :30036 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
 :30037 NAD/TEST.FUNC7 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
 :30038 -----
 :30039 ;079:
 :30040 ;=0*1
 :30041
 :30042 NAD/GROUP1
 :30043 -----
 :30044 ;182:

```

B  
C  
D  
E  
F  
G  
H  
I  
J  
K  
L  
M  
N  
O  
P  
Q  
R  
S  
T  
U  
V  
W  
X  
Y  
Z

```

:30045 :=0**
:30046 :GROUP1:
:30047 :
:30048 : BEN2/R80.FORMAT.H, ;WHEN THE R80 FORMAT BIT SETS..BRANCH
:30049 : NAD/GROUP1 ; TO NEXT WORD AND THEN BRANCH ON THE
:30050 : ; FUNCTION BITS
:30051 :-----
:30052 :186:
:30053 :=1**
:30054 :
:30055 : BEN0/F0,
:30056 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:30057 : BEN2/F2,
:30058 : NAD/GROUP1.F0
:30059 :-----
:30060 :028:
:30061 :=000
:30062 :GROUP1.F0:
:30063 :
:30064 :CLR.MATCH.1:
:30065 :
:30066 : UCON/CLR.MATCH, ;CLR THE MATCH CONDITION
:30067 : NAD/TEST.MATCH.1
:30068 :-----
:30069 :1F6:
:30070 :TEST.MATCH.1:
:30071 :
:30072 : BEN2/MISMATCH.L, ;IS MATCH ASSERTED?
:30073 : NAD/BEN2.1
:30074 :-----
:30075 :1A0:
:30076 :=0**
:30077 :BEN2.1:
:30078 :
:30079 : INCR.DAR/ON, ;INCR DAR
:30080 : NAD/XFER.REQ ;FLAG THE CPU
:30081 :-----
:30082 :1A4:
:30083 :=1**
:30084 :BEN2.2:
:30085 :
:30086 : INCR.DAR/ON, ;INCR DAR IF BEN2 CONDITION ASSERTED
:30087 : NAD/BEN2.1 ;INCR DAR AGAIN IF BEN2 COND ASSERTED
:30088 :-----
:30089 :1D8:
:30090 :XFER.REQ:
:30091 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:30092 : NAD/DIAG.WAIT
:30093 :-----
:30094 :1D7:
:30095 :DIAG.WAIT:
:30096 :
:30097 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:30098 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:30099 :-----

```



```

:30100 :159:
:30101 :=*0*
:30102 :DIAG.WAIT1:
:30103 :
:30104 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:30105 : ; COMMAND FROM THE CPU
:30106 :-----
:30107 :15B:
:30108 :=*1*
:30109 :DIAG.WAIT2:
:30110 :
:30111 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:30112 : ; WAIT SOME MORE
:30113 :-----
:30114 :WAIT.IDC.T35.1:
:30115 : SKIP IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ FROM IDC
:30116 : JMP [IDC.DONE.T35.1] ;XFER REQ SET
:30117 : JMP [WAIT.IDC.T35.1] ;WAIT SOME MORE
:30118 :
:30119 :IDC.DONE.T35.1:
:30120 : MOV LS[SETCSR.F0] TO WR[0] ;CLEAR CSR F<2:0>
:30121 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:30122 : MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
:30123 :
:30124 :LOOP.T35.1:
:30125 :;SET UP THE CSR TO BRANCH TO THE CORRECT IDC ROUTINE THAT WILL SET OPI
:30126 :
:30127 : MOV LS[SETCSR.F7] TO WR[0] ;SET UP THE CSR
:30128 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:30129 : ; MAINT=0,CRDY=0,F<2:0>=111
:30130 :;CONTINUE TO BRANCH
:30131 : MOV LS[SETCSR.R80] TO WR[0]
:30132 : MOV LS[SETCSR.F7] TO WR[1]
:30133 : BIS WR[1] TO WR[0]
:30134 : MOV LS[SETCSR.CRDY] TO WR[1] ;SET CRDY TO ALLOW OPI.L
:30135 : BIS WR[1] TO WR[0]
:30136 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:30137 : ;F<2:0>=111
:30138 :;SET.OPI
:30139 :-----
:30140 :-----
:30141 : *****
:30142 : * IDC *
:30143 : *****
:30144 :-----
:30145 :018:
:30146 :=000
:30147 :DIAG.IDLE:
:30148 :
:30149 : BEN0/F0,
:30150 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:30151 : BEN2/F2,
:30152 : NAD/DIAG.IDLE
:30153 :-----
:30154 :01F:

```

U 18DA, DB00,1A  
 U 18DB, 898D,D4  
 U 18DC, 098D,A4

U 18DD, B700,15  
 U 18DE, 17A0,15  
 U 18DF, 90FA,15

U 18E0, 370E,15  
 U 18E1, 17A0,15

U 18E2, 379E,15  
 U 18E3, B70E,95  
 U 18E4, AEC2,15  
 U 18E5, 3712,95  
 U 18E6, AEC2,15  
 U 18E7, 17A0,15

```

:30155 : =111
:30156 :
:30157 : BEN0/CRDY.L,
:30158 : BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:30159 : NAD/TEST.FUNC7 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:30160 : -----
:30161 : 079:
:30162 : =0*1
:30163 :
:30164 : NAD/GROUP1
:30165 : -----
:30166 : 182:
:30167 : =0**
:30168 : GROUP1:
:30169 :
:30170 : BEN2/R80.FORMAT.H, ;WHEN THE R80 FORMAT BIT SETS..BRANCH
:30171 : NAD/GROUP1 ; TO NEXT WORD AND THEN BRANCH ON THE
:30172 : ; FUNCTION BITS
:30173 : -----
:30174 : 186:
:30175 : =1**
:30176 :
:30177 : BEN0/F0,
:30178 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:30179 : BEN2/F2,
:30180 : NAD/GROUP1.F0
:30181 : -----
:30182 : 02F:
:30183 : =111
:30184 :
:30185 : UCMD/SET.OPI, ;SET OPI FOR CSR OPI TEST
:30186 :
:30187 : FUNC/SET.SSE, ;SET SKIP SECTOR ERROR
:30188 : NAD/OPI.BR.TST ;USED FOR OPI.L BRANCH TEST...DOES NOT AFFECT
:30189 : ; SET SSE OR SET.OPI
:30190 : -----
:30191 : 02B:
:30192 : =011
:30193 : OPI.BR.TST:
:30194 :
:30195 : BEN2/OPI.L, ;IS OPI ASSERTED?
:30196 : NAD/BEN2.1
:30197 : -----
:30198 : 1A0:
:30199 : =0**
:30200 : BEN2.1:
:30201 :
:30202 : INCR.DAR/ON, ;INCR DAR
:30203 : NAD/XFER.REQ ;FLAG THE CPU
:30204 : -----
:30205 : 1A4:
:30206 : =1**
:30207 : BEN2.2:
:30208 :
:30209 : INCR.DAR/ON, ;INCR DAR IF BEN2 CONDITION ASSERTED

```

```

30210 : NAD/BEN2.1 ; INCR DAR AGAIN IF BEN2 COND ASSERTED
30211 : -----
30212 : 1D8:
30213 : XFER.REQ:
30214 : UCMD/SET.XFER.RQST, ; SEND XFER REQ TO THE CPU
30215 : NAD/DIAG.WAIT
30216 : -----
30217 : 1D7:
30218 : DIAG.WAIT:
30219 :
30220 : BEN1/XFER.REQ, ; WAIT FOR THE CPU TO ISSUE XFER GRANT
30221 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
30222 : -----
30223 : 159:
30224 : =*0*
30225 : DIAG.WAIT1:
30226 :
30227 : NAD/DIAG.IDLE ; RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
30228 : ; COMMAND FROM THE CPU
30229 : -----
30230 : 158:
30231 : =*1*
30232 : DIAG.WAIT2:
30233 :
30234 : NAD/DIAG.WAIT ; XFER GRANT HAS NOT BEEN ISSUED....
30235 : ; WAIT SOME MORE
30236 : -----
30237 : WAIT.IDC.T35.2:
30238 : SKIP.IF[NO.FAST.INTERRUPT] ; WAIT FOR XFER REQ FROM IDC
30239 : JMP [IDC.DONE.T35.2] ; XFER REQ SET
30240 : JMP [WAIT.IDC.T35.2] ; WAIT SOME MORE
30241 :
30242 : IDC.DONE.T35.2:
30243 : MOV LS[SETCSR.F0] TO WR[0] ; CLEAR CSR F<2:0>
30244 : MOV LS[SETCSR.CRDY] TO WR[1] ; SET CRDY
30245 : BIS WR[1] TO WR[0]
30246 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ; WRITE THE CSR
30247 : MISC2 [TRANSFER.GRANT] ; SEND GRANT TO CLR THE XFER REQ
30248 :
30249 : READ.CSR.T35.1:
30250 : MISC [SET.PORT.I.0] ; SELECT THE IDC
30251 : PORT.READ PORT.ADRS[CSR] ; SET UP TO READ THE CSR
30252 : MOV PORT TO LS[PORT0] ; READ THE DATA FROM CSR AND
30253 : ; SAVE IT IN LS
30254 : MOV LS[PORT0] TO WR[0] ; SET UP RECEIVED DATA
30255 : MOV LS[#9400] TO WR[1] ; SET UP EXPECTED DATA
30256 : JSR [CHECK.RESULT] ; CHECK THE DATA
30257 : JMP [LOOP.T35.1] ; LOOP HERE ERROR & LOE IS SET
30258 :
30259 :
30260 : INC LS[ERROR.NUMBER] ; SET UP FOR ERROR 2
30261 :
30262 : MOV LS[#FFFFFFBFF] TO WR[0] ; SET UP THE MASK TO CHECK
30263 : MOV WR[0] TO LS[ERROR.MASK] ; BIT <10>
30264 :

```

U 18E8, DB00,1A  
 U 18E9, 898E,B4  
 U 18EA, 898E,84

U 18EB, B700,15  
 U 18EC, 3712,95  
 U 18ED, AEC2,15  
 U 18EE, 17A0,15  
 U 18EF, 90FA,15

U 18F0, 9090,15  
 U 18F1, 9780,15  
 U 18F2, 3B32,15

U 18F3, 3732,15  
 U 18F4, B7C8,95  
 U 18F5, 0869,3C  
 U 18F6, 098E,04

U 18F7, FF82,15

U 18F8, 37E0,15  
 U 18F9, 3E8A,15

```

;30265 LOOP.T35.2:
U 18FA, 3678,15 :30266 MOV LS[BIT28] TO WR[0] ;CLR CSR CRDY
;30267 ; BIT 28 PREVENTS IDC TIMEOUTS
U 18FB, 17A0,15 :30268 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR TO CLR CRDY
;30269
;30270 READ.CSR.T35.2:
U 18FC, 9090,15 :30271 MISC [SET.PORT.I.0] ;SELECT THE IDC
U 18FD, 9780,15 :30272 PORT.READ PORT.ADRS[CSR] ;SET UP TO READ THE CSR
U 18FE, 3B32,15 :30273 MOV PORT TO LS[PORT0] ;READ THE DATA FROM CSR AND
;30274 ; SAVE IT IN LS
U 18FF, 3732,15 :30275 MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
U 1900, 2F82,95 :30276 CLR WR[1] ;SET UP EXPECTED DATA
U 1901, 0869,3C :30277 JSR [CHECK.RESULT] ;CHECK THE DATA
U 1902, 898F,A4 :30278 JMP [LOOP.T35.2] ;LOOP HERE IF ERR & LOE IS SET
;30279
U 1903, FF82,15 :30280 INC LS[ERROR.NUMBER] ;SET UP FOR ERROR 3
;30281
;30282 ;SET OPI
;30283
;30284
U 1904, 370E,15 :30285 MOV LS[SETCSR.F7] TO WR[0] ;SET UP THE CSR
U 1905, 17A0,15 :30286 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
;30287 ; MAINT=0,CRDY=0,F<2:0>=111
;30288 ;CONTINUE TO BRANCH
U 1906, 379E,15 :30289 MOV LS[SETCSR.R80] TO WR[0]
U 1907, B70E,95 :30290 MOV LS[SETCSR.F7] TO WR[1]
U 1908, AEC2,15 :30291 BIS WR[1] TO WR[0]
U 1909, 3712,95 :30292 MOV LS[SETCSR.CRDY] TO WR[1] ;SET CRDY TO ALLOW OPI.L
U 190A, AEC2,15 :30293 BIS WR[1] TO WR[0]
U 190B, 17A0,15 :30294 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
;30295 ;F<2:0>=111
;30296 ;SET.OPI
;30297 -----
;30298 -----
;30299 :
;30300 : *****
;30301 : * IDC *
;30302 : *****
;30303 :-----
;30304 :C18:
;30305 :=000
;30306 :DIAG.IDLE:
;30307 :
;30308 : BEN0/F0,
;30309 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
;30310 : BEN2/F2,
;30311 : NAD/DIAG.IDLE
;30312 :-----
;30313 :01F:
;30314 :=111
;30315 :
;30316 : BEN0/CRDY.L,
;30317 : BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
;30318 : NAD/TEST.FUNC7 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
;30319 :-----
;30319 :079:

```

```

:30320 :=0*1
:30321 :
:30322 : NAD/GROUP1
:30323 :-----
:30324 :182:
:30325 :=0**
:30326 :GROUP1:
:30327 :
:30328 : BEN2/R80.FORMAT.H, ;WHEN THE R80 FORMAT BIT SETS..BRANCH
:30329 : NAD/GROUP1 ; TO NEXT WORD AND THEN BRANCH ON THE
:30330 : ; FUNCTION BITS
:30331 :-----
:30332 :186:
:30333 :=1**
:30334 :
:30335 : BEN0/F0,
:30336 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:30337 : BEN2/F2,
:30338 : NAD/GROUP1.F0
:30339 :-----
:30340 :02F:
:30341 :=111
:30342 :
:30343 : UCMD/SET.OPI, ;SET OPI FOR CSR OPI TEST
:30344 :
:30345 : FUNC/SET.SSE, ;SET SKIP SECTOR ERROR
:30346 : NAD/OPI.BR.TST ;USED FOR OPI.L BRANCH TEST...DOES NOT AFFECT
:30347 : ; SET SSE OR SET.OPI
:30348 :-----
:30349 :02B:
:30350 :=011
:30351 :OPI.BR.TST:
:30352 :
:30353 : BEN2/OPI.L, ;IS OPI ASSERTED?
:30354 : NAD/BEN2.1
:30355 :-----
:30356 :1A0:
:30357 :=0**
:30358 :BEN2.1:
:30359 :
:30360 : INCR.DAR/ON, ;INCR DAR
:30361 : NAD/XFER.REQ ;FLAG THE CPU
:30362 :-----
:30363 :1A4:
:30364 :=1**
:30365 :BEN2.2:
:30366 :
:30367 : INCR.DAR/ON, ;INCR DAR IF BEN2 CONDITION ASSERTED
:30368 : NAD/BEN2.1 ;INCR DAR AGAIN IF BEN2 COND ASSERTED
:30369 :-----
:30370 :1D8:
:30371 :XFER.REQ:
:30372 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:30373 : NAD/DIAG.WAIT
:30374 :-----

```

```

30375 :107:
30376 :DIAG.WAIT:
30377 :
30378 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
30379 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
30380 :-----
30381 :159:
30382 :=*0*
30383 :DIAG.WAIT1.
30384 :
30385 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
30386 : ; COMMAND FROM THE CPU
30387 :-----
30388 :15B:
30389 :=*1*
30390 :DIAG.WAIT2:
30391 :
30392 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
30393 : ; WAIT SOME MORE
30394 :-----
30395 :WAIT.IDC.T35.3:
30396 : SKIP IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ FROM IDC
30397 : JMP [IDC.DONE.T35.3] ;XFER REQ SET
30398 : JMP [WAIT.IDC.T35.3] ;WAIT SOME MORE
30399 :-----
30400 :IDC.DONE.T35.3:
30401 : MOV LS[SETCSR.F0] TO WR[0] ;CLEAR CSR ?<2:0>
30402 : MOV LS[SETCSR.CRDY] TO WR[1] ;SET CRDY
30403 : BIS WR[1] TO WR[0]
30404 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
30405 : MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
30406 :-----
30407 :LOOP.T35.3:
30408 : PORT.CONTROL [CLEAR.IDC] ;INIT THE IDC
30409 : PORT.CONTROL [CLEAR.IDC]
30410 : PORT.CONTROL [CLEAR.IDC]
30411 :-----
30412 :READ.CSR.T35.3:
30413 : MISC [SET.PORT.1.0] ;SELECT THE IDC
30414 : PORT.READ PORT.ADRS[CSR] ;SET UP TO READ THE CSR
30415 : MOV PORT TO LS[PORT0] ;READ THE DATA FROM CSR AND
30416 : ; SAVE IT IN LS
30417 : MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
30418 : CLR WR[1] ;SET UP EXPECTED DATA
30419 : JSR [CHECK.RESULT] ;CHECK THE DATA
30420 : JMP [LOOP.T35.3] ;LOOP HERE IF ERR & LOE IS SET
30421 :-----
30422 : JSR [CLEANUP] ;CLEANUP CONDITIONS SET BY
30423 : ; THIS TEST
30424 : ;END TEST
30425 :END.T35:
30426 :
30427 :

```

U 190C, DB00,1A  
 U 190D, 0990,F4  
 U 190E, 0990,C4

U 190F, B700,15  
 U 1910, 3712,95  
 U 1911, AEC2,15  
 U 1912, 17A0,15  
 U 1913, 90FA,15

U 1914, 17CC,15  
 U 1915, 17CC,15  
 U 1916, 17CC,15

U 1917, 9090,15  
 U 1918, 9780,15  
 U 1919, 3B32,15

U 191A, 3732,15  
 U 191B, 2F82,95  
 U 191C, 0869,3C  
 U 191D, 0991,44

U 191E, 8875,3C

30428 : PAGE "Test 36 OPERATION INCOMPLETE BRANCH TEST (M8388 IDC MODULE)"

30429 : \*\*  
 30430 : Test Description:

30431 : This test is designed to insure that the BEN2/OPI.L branch condition  
 30432 : operates correctly. This is done by first setting OPI.L by issuing a  
 30433 : SET.OPI.L from the IDC microcode. BEN2/OPI.L is then selected by the  
 30434 : IDC microcode. If OPI.L is asserted the DAR will be incremented  
 30435 : once and if OPI.L is not asserted the DAR will be incremented twice.  
 30436 : The CPU will then read and check the the DAR = 1.  
 30437 : CSR <CRDY> will then be cleared which will clear OPI.L. BEN2/OPI.L  
 30438 : will again be selected. If OPI.L is not asserted, the DAR will be  
 30439 : incremented twice and if OPI.L is asserted the DAR will be incremented  
 30440 : once. The CPU will then read and check the DAR = 2.

30441 :  
 30442 : Assumptions:

30443 : It is assumed that the CSR OPERATIONS INCOMPLETE BITS TEST has  
 30444 : previously been run successfully.

30445 :  
 30446 : Test Steps:

- 30447 : 1. Set up error mask, error number and module number in LS
- 30448 : (for error printouts) and clear the previous error number in LS.
- 30449 : 2. Force the IDC to DIAG.IDLE.
- 30450 : 3. Clear the DAR.
- 30451 : 4. Set up the CSR to branch to the correct IDC routine.

30452 : \*\*\*\*\*  
 30453 : \* IDC \*  
 30454 : \*\*\*\*\*

- 30455 : IDC1. Execute an instruction with UCM <2:0> = 2, SET.OPI.
- 30456 : IDC2. Select BEN2/OPI L.
- 30457 : IDC3. Increment the DAR once if OPI L is asserted.
- 30458 : Increment the DAR twice if OPI L is NOT asserted.
- 30459 : IDC4. Send XFER REQ to flag the CPU.

- 30460 : 5. Wait for XFER REQ from IDC.
- 30461 : 6. Clear CSR F<2:0> and send XFER GRANT to clear the XFER REQ.
- 30462 : 7. Check the DAR = 1.
- 30463 : 8. Increment the error number.
- 30464 : 9. Clear the DAR.
- 30465 : 10. Clear CSR <CRDY> to clear OPI L.
- 30466 : 11. Set up the CSR to branch to the IDC routine that will select
- 30467 : BEN2/OPI L.

30468 : \*\*\*\*\*  
 30469 : \* IDC \*  
 30470 : \*\*\*\*\*

- 30471 : IDC6. Select BEN2/OPI L.
- 30472 : IDC7. Increment the DAR once if OPI L is asserted.
- 30473 : Increment the DAR twice if OPI L is NOT asserted.
- 30474 : IDC8. Send XFER REQ to flag the CPU.

30483 :  
 30484 : 12. Wait for XFER REQ from IDC.  
 30485 : 13. Clear CSR F<2:0> and send XFER GRANT to clear the XFER REQ.  
 30486 : 14. Check the DAR = 2.  
 30487 : 15. Clear IDC.  
 30488 : 16. End test.  
 30489 :

30490 : Errors:

30491 :  
 30492 : Printout:  
 30493 : EXPECTED RECEIVED  
 30494 : DAR DAR  
 30495 :

30496 : ERROR 1 - BEN2/OPI L failed when asserted.

30497 : ERROR 2 - BEN2/OPI L failed when NOT asserted.

30498 : Debug:

30499 : ERROR 1 -

30500 : The DAR is cleared. The IDC will branch to an instruction with  
 30501 : UCMD <2:0> = 010, SET.OPI.L. The IDC microcode will then select  
 30502 : BEN2/OPI.L. (BEN2 S2 H = L, BEN2 S1 H = H, BEN2 S0 H = H). OPI L  
 30503 : will be asserted so NUA 2 H = L and the microcode will branch to  
 30504 : a series of instructions that will increment the DAR once.  
 30505 :

30506 : ERROR 2 -

30507 : OPI L will be asserted. The DAR is cleared. A PORT.WRITE PORT.ADRS[CSR]  
 30508 : will be issued with CRDY (bit 7) cleared. This instruction will be  
 30509 : decoded by the PORT RD/WR DECODE PAL and will assert WRITE CSR L.  
 30510 : WRITE CSR L = L and BUS IO 7 H = L are inputs to the CONTROL STATUS  
 30511 : REG PAL and on the next P2 CLOCK L will deassert OPI L. (OPI L = H).  
 30512 : The IDC microcode will branch to an instruction that will select  
 30513 : BEN2/OPI.L (BEN2 S2 H = L, BEN2 S1 H = H, BEN2 S0 H = H).  
 30514 : OPI L will be H so NUA 2 H = H and the IDC microcode will branch to a  
 30515 : series of instructions that will increment the DAR twice.  
 30516 :

30517 : --

30518 : \*\*\*\*\*

30519 : T.36:

|                 |       |                                 |                               |
|-----------------|-------|---------------------------------|-------------------------------|
| U 191F, B65E,15 | 30520 | MOV LS[BEGIN.TEST] TO WR[0]     | : SET BIT 15 IN WR[0] FOR     |
|                 | 30521 |                                 | : CONTROL AND STATUS WORD     |
| U 1920, 3E80,15 | 30522 | MOV WR[0] TO LS[CONTROL.STATUS] | : SET BIT 15 IN CONTROL AND   |
|                 | 30523 |                                 | : STATUS WORD - BIT 15        |
|                 | 30524 |                                 | : INDICATES START OF TEST TO  |
|                 | 30525 |                                 | : CONSOLE                     |
| U 1921, 10E0,15 | 30526 | MISC [SET.CP.ATTN]              | : ISSUE CPU ATTN TO CONSOLE   |
|                 | 30527 | WAIT.T36.0:                     |                               |
| U 1922, 0992,24 | 30528 | JMP [WAIT.T36.0]                | : LOOP HERE WAITING FOR       |
|                 | 30529 |                                 | : CONSOLE TO RESPOND          |
| U 1923, 65A0,15 | 30530 | CLR LS[PREVIOUS.ERROR]          | : CLEAR PREVIOUS ERROR NUMBER |
|                 | 30531 |                                 | : IN LS                       |
| U 1924, B64A,15 | 30532 | MOV LS[IDC] TO WR[0]            | : STORE MODULE CODE IN LS TO  |
|                 | 30533 |                                 |                               |
|                 | 30534 |                                 |                               |
|                 | 30535 |                                 |                               |
|                 | 30536 |                                 |                               |
|                 | 30537 |                                 |                               |



J 16

22-ENKCF-1.4 : ENKCF.MIC Test 36 OPERATION 15-MAR-83 Fiche 3 Frame J16 Sequence 616  
: ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40 Page 610  
: ENKCF.MIC Test 36 OPERATION INCOMPLETE BRANCH TEST (M8388 IDC MODULE)

```

U 1925, 3E8C,15 :30538 MOV WR[0] TO LS[MODULE.NUM] ; INDICATE IDC MODULE
:30539
:30540
U 1926, B640,15 :30541 MOV LS[#1] TO WR[0] ;SET WRO = 1
U 1927, BE82,15 :30542 MOV WR[0] TO LS[ERROR.NUMBER] ;SET UP ERROR NUMBER = 1 IN LS
:30543
U 1928, 3776,15 :30544 MOV LS[DAR.MASK] TO WR[0]
U 1929, 3E8A,15 :30545 MOV WR[0] TO LS[ERROR.MASK] ;SET UP MASK TO CHECK <18:0>
:30546
U 192A, 8872,EC :30547 JSR [DIAG.CODE] ;INIT THE IDC AND FORCE IT TO
:30548 ; DIAG.IDLE
:30549
:30550 LOOP.T36.1:
U 192B, 2F80,15 :30551 CLR WR[0]
U 192C, 97A4,15 :30552 PORT.WRITE PORT.ADRS[DAR] WITH WR[0] ;CLEAR THE DAR
:30553
:30554 ;SET UP THE CSR TO BRANCH TO THE CORRECT IDC ROUTINE THAT WILL SET OPI
:30555
U 192D, 370E,15 :30556 MOV LS[SETCSR.F7] TO WR[0] ;SET UP THE CSR
U 192E, 17A0,15 :30557 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:30558 ; MAINT=0,CRDY=0,F<2:0>=111
:30559 ;CONTINUE TO BRANCH
U 192F, 379E,15 :30560 MOV LS[SETCSR.R80] TO WR[0]
U 1930, B70E,95 :30561 MOV LS[SETCSR.F7] TO WR[1]
U 1931, AEC2,15 :30562 BIS WR[1] TO WR[0]
U 1932, 3712,95 :30563 MOV LS[SETCSR.CRDY] TO WR[1] ;SET CRDY TO ALLOW OPI.L
U 1933, AEC2,15 :30564 BIS WR[1] TO WR[0]
U 1934, 17A0,15 :30565 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:30566 ;F<2:0>=111
:30567 ;SET.OPI AND THEN SELECT BEN2/OPI.L
:30568 -----
:30569 -----
:30570 ;*****
:30571 ; IDC ;
:30572 ;*****
:30573 -----
:30574 :018:
:30575 :=000
:30576 DIAG.IDLE:
:30577
:30578 BEN0/F0,
:30579 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:30580 BEN2/F2,
:30581 NAD/DIAG.IDLE
:30582 -----
:30583 :01F:
:30584 :=111
:30585
:30586 BEN0/CRDY.L,
:30587 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:30588 NAD/TEST.FUNC7 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:30589 -----
:30590 :079:
:30591 :=0*1
:30592

```

```

30593 : NAD/GROUP1
30594 : -----
30595 : 182:
30596 : =0**
30597 : GROUP1:
30598 :
30599 : BEN2/R80.FORMAT.H, ; WHEN THE R80 FORMAT BIT SETS..BRANCH
30600 : NAD/GROUP1 ; TO NEXT WORD AND THEN BRANCH ON THE
30601 : ; FUNCTION BITS
30602 : -----
30603 : 186:
30604 : =1**
30605 :
30606 : BEN0/F0,
30607 : BEN1/F1, ; BRANCH ON THE CSR F<2:0> BITS
30608 : BEN2/F2,
30609 : NAD/GROUP1.F0
30610 : -----
30611 : 02F:
30612 : =111
30613 :
30614 : UCMD/SET.OPI, ; SET OPI FOR CSR OPI TEST
30615 :
30616 : FUNC/SET.SSE, ; SET SKIP SECTOR ERROR
30617 : NAD/OPI.BR.TST ; USED FOR OPI.L BRANCH TEST...DOES NOT AFFECT
30618 : ; SET SSE OR SET.OPI
30619 : -----
30620 : 02B:
30621 : =011
30622 : OPI.BR.TST:
30623 :
30624 : BEN2/OPI.L, ; IS OPI ASSERTED?
30625 : NAD/BEN2.1
30626 : -----
30627 : 1A0:
30628 : =0**
30629 : BEN2.1:
30630 :
30631 : INCR.DAR/ON, ; INCR DAR
30632 : NAD/XFER.REQ ; FLAG THE CPU
30633 : -----
30634 : 1A4:
30635 : =1**
30636 : BEN2.2:
30637 :
30638 : INCR.DAR/ON, ; INCR DAR IF BEN2 CONDITION ASSERTED
30639 : NAD/BEN2.1 ; INCR DAR AGAIN IF BEN2 COND ASSERTED
30640 : -----
30641 : 1D8:
30642 : XFER.REQ:
30643 : UCMD/SET.XFER.RQST, ; SEND XFER REQ TO THE CPU
30644 : NAD/DIAG.WAIT
30645 : -----
30646 : 1D7:
30647 : DIAG.WAIT:

```

```

:30648 :
:30649 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:30650 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:30651 :-----
:30652 :159:
:30653 :=*0*
:30654 :DIAG.WAIT1:
:30655 :
:30656 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:30657 : ; COMMAND FROM THE CPU
:30658 :-----
:30659 :15B:
:30660 :=*1*
:30661 :DIAG.WAIT2:
:30662 :
:30663 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:30664 : ; WAIT SOME MORE
:30665 :-----
:30666 :WAIT.IDC.T36.1:
U 1935, DB00,1A :30667 :SKIP.[IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ FROM IDC
U 1936, 8993,84 :30668 :JMP [IDC.DONE.T36.1] ;XFER REQ SET
U 1937, 0993,54 :30669 :JMP [WAIT.IDC.T36.1] ;WAIT SOME MORE
:30670 :
:30671 :IDC.DONE.T36.1:
U 1938, B700,15 :30672 :MOV LS[SETCSR.F0] TO WR[0] ;CLEAR THE FUNC BITS
U 1939, 17A0,15 :30673 :PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ; WRITE THE DATA TO IDC CSR
U 193A, 90FA,15 :30674 :MISC2 [TRANSFER.GRANT] ;SEND TRANSFER GRANT
:30675 :
:30676 :READ.DAR.T36.1:
U 193B, 9090,15 :30677 :MISC [SET.PORT.I.0] ;SELECT THE IDC
U 193C, 1784,15 :30678 :PORT.READ PORT.ADRS[DAR] ;SET UP TO READ THE DAR
U 193D, 3B32,15 :30679 :MOV PORT TO LS[PORT0] ;READ THE DATA FROM DAR AND
:30680 : ; SAVE IT IN LS
U 193E, 3732,15 :30681 :MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
U 193F, 3640,95 :30682 :MOV LS[#1] TO WR[1] ;SET UP EXPECTED DATA
U 1940, 0869,3C :30683 :JSR [CHECK.RESULT] ;CHECK THE DATA
U 1941, 0992,B4 :30684 :JMP [LOOP.T36.1] ;LOOP HERE IF ERR & LOE IS SET
:30685 :
U 1942, FF82,15 :30686 :INC LS[ERROR.NUMBER] ;ERROR 2
:30687 :
:30688 :LOOP.T36.2:
U 1943, 2F80,15 :30689 :CLR WR[0]
U 1944, 97A4,15 :30690 :PORT.WRITE PORT.ADRS[DAR] WITH WR[0] ;CLEAR THE DAR
U 1945, 4778,15 :30691 :BIS LS[BIT28] TO WR[0] ;SET THE TIMEOUT INHIBIT BIT
:30692 : ; BEFORE WRITING TO THE CSR
U 1946, 17A0,15 :30693 :PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;CLR CSR <CRDY> TO CLR OPI.L
:30694 :
:30695 :;SET UP THE CSR TO SELECT BEN2/OPI.L
:30696 :
U 1947, 370E,15 :30697 :MOV LS[SETCSR.F7] TO WR[0] ;SET UP THE CSR
U 1948, 17A0,15 :30698 :PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:30699 : ; MAINT=0,CRDY=0,F<2:0>=111
:30700 :;CONTINUE TO BRANCH
U 1949, 379E,15 :30701 :MOV LS[SETCSR.R80] TO WR[0]
U 194A, 3706,95 :30702 :MOV LS[SETCSR.F3] TO WR[1]

```

U  
U  
U  
U  
U  
U  
U

```

:30758 :BEN2.1:
:30759 :
:30760 : INCR.DAR/ON, ; INCR DAR
:30761 : NAD/XFER.REQ ; FLAG THE CPU
:30762 :-----
:30763 :1A4:
:30764 :=1**
:30765 :BEN2.2:
:30766 :
:30767 : INCR.DAR/ON, ; INCR DAR IF BEN2 CONDITION ASSERTED
:30768 : NAD/BEN2.1 ; INCR DAR AGAIN IF BEN2 COND ASSERTED
:30769 :-----
:30770 :1D8:
:30771 :XFER.REQ:
:30772 : UCMD/SET.XFER.RQST, ; SEND XFER REQ TO THE CPU
:30773 : NAD/DIAG.WAIT
:30774 :-----
:30775 :1D7:
:30776 :DIAG.WAIT:
:30777 :
:30778 : BEN1/XFER.REQ, ; WAIT FOR THE CPU TO ISSUE XFER GRANT
:30779 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:30780 :-----
:30781 :159:
:30782 :=*0*
:30783 :DIAG.WAIT1:
:30784 :
:30785 : NAD/DIAG.IDLE ; RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:30786 : ; COMMAND FROM THE CPU
:30787 :-----
:30788 :15B:
:30789 :=*1*
:30790 :DIAG.WAIT2:
:30791 :
:30792 : NAD/DIAG.WAIT ; XFER GRANT HAS NOT BEEN ISSUED....
:30793 : ; WAIT SOME MORE
:30794 :-----
:30795 :WAIT.IDC.T36.2:
:30796 : SKIP.IF[NO.FAST.INTERRUPT] ; WAIT FOR XFER REQ FROM IDC
:30797 : JMP [IDC.DONE.T36.2] ; XFER REQ SET
:30798 : JMP [WAIT.IDC.T36.2] ; WAIT SOME MORE
:30799 :
:30800 :IDC.DONE.T36.2:
:30801 : MOV LS[SETCSR.F0] TO WR[0] ; CLEAR CSR F<2:0>
:30802 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ; WRITE THE CSR
:30803 : MISC2 [TRANSFER.GRANT] ; SEND GRANT TO CLR THE XFER REQ
:30804 :
:30805 :READ.DAR.T36.2:
:30806 : MISC [SET.PORT.1.0] ; SELECT THE IDC
:30807 : PORT.READ PORT.ADRS[DAR] ; SET UP TO READ THE DAR
:30808 : MOV PORT TO LS[PORT0] ; READ THE DATA FROM DAR AND
:30809 : ; SAVE IT IN LS
:30810 : MOV LS[PORT0] TO WR[0] ; SET UP RECEIVED DATA
:30811 : MOV LS[#2] TO WR[1] ; SET UP EXPECTED DATA
:30812 : JSR [CHECK.RESULT] ; CHECK THE DATA

```

U 194D, DB00,1A  
U 194E, 0995,04  
U 194F, 0994,D4

U 1950, B700,15  
U 1951, 17A0,15  
U 1952, 90FA,15

U 1953, 9090,15  
U 1954, 1784,15  
U 1955, 3B32,15

U 1956, 3732,15  
U 1957, B642,95  
U 1958, 0869,3C

D 1  
 ZZ-ENKCF-1.4 : ENKCF.MIC Test 36 OPERATION 15-MAR-83 Fiche 4 Frame D1 Sequence 621  
 : ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40 Page 615  
 : ENKCF.MIC Test 36 OPERATION INCOMPLETE BRANCH TEST (M8388 IDC MODULE)  
 U 1959, 8994,34 :30813 JMP [LOOP.T36.2] ;LOOP HERE IF ERR & LOE IS SET  
 :30814  
 U 195A, 8875,3C :30815 JSR [CLEANUP] ;CLEANUP THE CONDITIONS SET  
 :30816 ; BY THIS TEST  
 :30817 END.T36: ;END TEST  
 :30818  
 :30819  
 :30820

30821 PAGE "Test 37 CSR DATA LATE (DLT) BITS TEST (M8388 IDC MODULE)"

30822 ++

30823

30824 Test Description:

30825

30826 This test is designed to verify the CSR bits <12>, and CSR <15>  
 30827 set when the IDC issues a SET.DLT.

30828

30829

30830 Assumptions:

30831

30832 It is assumed that all previous tests have run successfully.

30833

30834 Test Steps:

30835

- 30836 1. Set up error mask, error number and module number in LS  
 30837 (for error printouts) and clear the previous error number in LS.
- 30838 2. Force the IDC microcode to DIAG.IDLE and INIT the IDC.
- 30839 3. Clear MATCH to assert MISMATCH L to test BIT <12>.
- 30840 4. Set up the CSR to branch to the correct IDC microcode routine.

30841

30842

30843

30844

30845

30846

30847

30848

30849

30850

30851

30852

30853

30854

30855

30856

30857

30858

30859

30860

30861

30862

30863

30864

30865

30866

30867

30868

30869

30870

30871

30872

30873

30874

30875

Errors:

Printout:

EXPECTED  
CSR

RECEIVED  
CSR

ERROR 1 - The CSR DLT bits <15> <12> did not set when the IDC issued SET.DLT.

ERROR 2 - CSR bit <12> did not clear when CSR <CRDY> was cleared.

ERROR 3 - CSR bit <12> did not clear when the IDC was INITed.

```

:30876 : Debug:
:30877 :
:30878 : ERROR 1 -
:30879 : The IDC will branch to an instruction that will CLR.MATCH. This
:30880 : will assert MISMATCH L at the DSK RD SEQ REG PAL. This is done so
:30881 : that CSR bit <12> can be tested.
:30882 : The IDC will then branch to an instruction with UCMD<2:0> = 011.
:30883 : These bits will be decoded by the CONTROL STATUS REG PAL and will
:30884 : assert DLT L, which then asserts OPI + DLT L. These signal is gated
:30885 : with MISMATCH L = L which will assert CSR <12>. OPI + DLT L is also
:30886 : gated with some other signals and in this time is inverted to become
:30887 : CSR <15>.
:30888 :
:30889 : ERROR 2 -
:30890 : CSR bit <12> is set. Then a PORT.WRITE PORT.ADRS[CSR] is issued by the
:30891 : CPU with bit <7> (CRDY) cleared. This instruction is decoded by the
:30892 : PORT RD/WR DECODE PAL and will assert WRITE CSR L. WRITE CSR L = L
:30893 : and BUS IO 07 H = L are inputs to the CONTROL STATUS REG PAL. This
:30894 : combination of inputs will clear DLT L which will then clear
:30895 : OPI + DLT L.
:30896 :
:30897 : ERROR 3 -
:30898 : CSR bit <12> is set.
:30899 : The CPU issues a series of PORT.CONTROL [CLEAR.IDC] instructions which
:30900 : are decoded by the PORT RD/WR DECODE PAL. This will assert CLEAR IDC L.
:30901 : At the next P2 CLOCK L, the INIT FF will assert INIT SYNC 1 H. At the
:30902 : next clock INIT H is asserted. INIT H is inverted to INIT L. This is
:30903 : an input to the CONTROL STATUS REG PAL and when INIT L asserts, DLT L
:30904 : will clear and this will clear OPI + DLT L.
:30905 : --
:30906 : *****
:30907 : T.37:

```

|                 |        |                                 |                                |
|-----------------|--------|---------------------------------|--------------------------------|
| U 195B, B65E,15 | :30908 | MOV LS[BEGIN.TEST] TO WR[0]     | :SET BIT 15 IN WR[0] FOR       |
|                 | :30909 |                                 | :CONTROL AND STATUS WORD       |
| U 195C, 3E80,15 | :30910 | MOV WR[0] TO LS[CONTROL.STATUS] | :SET BIT 15 IN CONTROL AND     |
|                 | :30911 |                                 | :STATUS WORD - BIT 15          |
|                 | :30912 |                                 | :INDICATES START OF TEST TO    |
|                 | :30913 |                                 | :CONSOLE                       |
| U 195D, 10E0,15 | :30914 | MISC [SET.CP.ATTN]              | :ISSUE CPU ATTN TO CONSOLE     |
|                 | :30915 | WAIT.T37.0:                     |                                |
| U 195E, 8995,E4 | :30916 | JMP [WAIT.T37.0]                | :LOOP HERE WAITING FOR         |
|                 | :30917 |                                 | :CONSOLE TO RESPOND            |
|                 | :30918 |                                 |                                |
| U 195F, 65A0,15 | :30919 | CLR LS[PREVIOUS.ERROR]          | :CLEAR PREVIOUS ERROR NUMBER   |
|                 | :30920 |                                 | :IN LS                         |
|                 | :30921 |                                 |                                |
| U 1960, B64A,15 | :30922 | MOV LS[IDC] TO WR[0]            | :STORE MODULE CODE IN LS TO    |
| U 1961, 3E8C,15 | :30923 | MOV WR[0] TO LS[MODULE.NUM]     | :INDICATE IDC MODULE           |
|                 | :30924 |                                 |                                |
|                 | :30925 |                                 |                                |
| U 1962, B640,15 | :30926 | MOV LS[#1] TO WR[0]             | :SET WRO = 1                   |
| U 1963, BE82,15 | :30927 | MOV WR[0] TO LS[ERROR.NUMBER]   | :SET UP ERROR NUMBER = 1 IN LS |
|                 | :30928 |                                 |                                |
| U 1964, B7CA,15 | :30929 | MOV LS[#FFFF6FFF] TO WR[0]      | :SET UP THE MASK TO CHECK      |
| U 1965, 3E8A,15 | :30930 | MOV WR[0] TO LS[ERROR.MASK]     | :BITS <15> <12>                |



|   |   |
|---|---|
| U | 1 |
| U | 1 |
| U | 1 |
|   |   |
| U | 1 |
| U | 1 |
| U | 1 |
|   |   |
| U | 1 |
| U | 1 |
| U | 1 |
|   |   |
| U | 1 |
| U | 1 |
| U | 1 |
| U | 1 |

```

:30986 : BEN2/F2,
:30987 : NAD/GROUP1.F0
:30988 :-----
:30989 : 028:
:30990 : =000
:30991 : GROUP1.F0:
:30992 :
:30993 : CLR.MATCH.1:
:30994 :
:30995 : UCON/CLR.MATCH, ;CLR THE MATCH CONDITION
:30996 : NAD/TEST.MATCH.1
:30997 :-----
:30998 : 1F6:
:30999 : TEST.MATCH.1:
:31000 :
:31001 : BEN2/MISMATCH.L, ;IS MATCH ASSERTED?
:31002 : NAD/BEN2.1
:31003 :-----
:31004 : 1A0:
:31005 : =0**
:31006 : BEN2.1:
:31007 :
:31008 : INCR.DAR/ON, ;INCR DAR
:31009 : NAD/XFER.REQ ;FLAG THE CPU
:31010 :-----
:31011 : 1A4:
:31012 : =1**
:31013 : BEN2.2:
:31014 :
:31015 : INCR.DAR/ON, ;INCR DAR IF BEN2 CONDITION ASSERTED
:31016 : NAD/BEN2.1 ;INCR DAR AGAIN IF BEN2 COND ASSERTED
:31017 :-----
:31018 : 1D8:
:31019 : XFER.REQ:
:31020 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:31021 : NAD/DIAG.WAIT
:31022 :-----
:31023 : 1D7:
:31024 : DIAG.WAIT:
:31025 :
:31026 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:31027 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:31028 :-----
:31029 : 159:
:31030 : =*0*
:31031 : DIAG.WAIT1:
:31032 :
:31033 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:31034 : ; COMMAND FROM THE CPU
:31035 :-----
:31036 : 15B:
:31037 : =*1*
:31038 : DIAG.WAIT2:
:31039 :
:31040 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....

```

```

31041 : ; WAIT SOME MORE
31042 : -----
31043 WAIT.IDC.T37.1:
31044 SKIP.IF[NO.FAST.INTERUPT] ;WAIT FOR XFER REQ FROM IDC
31045 JMP [IDC.DONE.T37.1] ;XFER REQ SET
31046 JMP [WAIT.IDC.T37.1] ;WAIT SOME MORE
31047 :
31048 IDC.DONE.T37.1:
31049 MOV LS[SETCSR.F0] TO WR[0] ;CLEAR CSR F<2:0>
31050 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
31051 MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
31052 :
31053 :
31054 ;SET UP THE CSR TO BRANCH TO THE ROUTINE THAT WILL SET.DLT
31055 LOOP.T37.1:
31056 MOV LS[SETCSR.F6] TO WR[0] ;SET UP THE CSR
31057 MOV LS[SETCSR.MAINT] TO WR[1]
31058 BIS WR[1] TO WR[0]
31059 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
31060 ;MAINT=1,CRDY=0,F<2:0>=110
31061 :SET.DLT
31062 : -----
31063 : -----
31064 : *****
31065 : * IDC *
31066 : *****
31067 : -----
31068 :018:
31069 :=000
31070 DIAG.IDLE:
31071 :
31072 BEN0/F0,
31073 BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
31074 BEN2/F2,
31075 NAD/DIAG.IDLE
31076 : -----
31077 :01E:
31078 :=110
31079 :
31080 BEN0/CRDY.L,
31081 BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
31082 NAD/TEST.FUNC6 ;DEPENDENT ON CSR <CRDY> AND <MAINT>
31083 : -----
31084 :077:
31085 :=1*1
31086 TEST.F6:
31087 :
31088 : ;DO NOT INCR DAR BETWEEN HERE AND
31089 : ; DIAG.IDLE OR DIAG.WAIT
31090 LOAD.LOOP.CNTR.[00], ;LOAD LOOP CNTR WITH 00
31091 :
31092 UCMD/SET.DLT, ;SET DLT FOR DLT TEST
31093 :
31094 FUNC/SET.INT, ;SET INT FOR INTERRUPT TEST
31095 :

```

```

:31096 : NAD/XFER.REQ ;SEND XFER REQ TO FLAG THE CPU
:31097 :-----
:31098 :1D8:
:31099 :XFER.REQ:
:31100 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:31101 : NAD/DIAG.WAIT
:31102 :-----
:31103 :1D7:
:31104 :DIAG.WAIT:
:31105 :
:31106 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:31107 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:31108 :-----
:31109 :159:
:31110 :=*0*
:31111 :DIAG.WAIT1:
:31112 :
:31113 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:31114 : ; COMMAND FROM THE CPU
:31115 :-----
:31116 :15B:
:31117 :=*1*
:31118 :DIAG.WAIT2:
:31119 :
:31120 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:31121 : ; WAIT SOME MORE
:31122 :-----
:31123 :WAIT.IDC.T37.2:
:31124 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ FROM IDC
:31125 : JMP [IDC.DONE.T37.2] ;XFER REQ SET
:31126 : JMP [WAIT.IDC.T37.2] ;WAIT SOME MORE
:31127 :-----
:31128 :IDC.DONE.T37.2:
:31129 : MOV LS[SETCSR.F0] TO WR[0] ;CLEAR CSR F<2:0>
:31130 : MOV LS[SETCSR.CRDY] TO WR[1] ;SET CRDY SO DLT DOES NOT CLR
:31131 : BIS WR[1] TO WR[0]
:31132 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:31133 : MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
:31134 :-----
:31135 :READ.CSR.T37.1:
:31136 : MISC [SET.PORT.1.0] ;SELECT THE IDC
:31137 : PORT.READ PORT.ADRS[CSR] ;SET UP TO READ THE CSR
:31138 : MOV PORT TO LS[PORT0] ;READ THE DATA FROM CSR AND
:31139 : ; SAVE IT IN LS
:31140 : MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
:31141 : MOV LS[#9000] TO WR[1] ;SET UP EXPECTED DATA
:31142 : JSR [CHECK.RESULT] ;CHECK THE DATA
:31143 : JMP [LOOP.T37.1] ;LOOP HERE IF ERR & LOE IS SET
:31144 :
:31145 : INC LS[ERROR.NUMBER] ;SET UP FOR ERROR 2
:31146 :
:31147 : MOV LS[#FFFFFFF] TO WR[0] ;SET UP THE MASK TO CHECK
:31148 : MOV WR[0] TO LS[ERROR.MASK] ; BIT <12>
:31149 :
:31150 :LOOP.T37.2:

```

U 1977, DB00,1A  
U 1978, 8997,A4  
U 1979, 0997,74

U 197A, B700,15  
U 197B, 3712,95  
U 197C, AEC2,15  
U 197D, 17A0,15  
U 197E, 90FA,15

U 197F, 9090,15  
U 1980, 9780,15  
U 1981, 3B32,15

U 1982, 3732,15  
U 1983, 37CC,95  
U 1984, 0869,3C  
U 1985, 8997,34

U 1986, FF82,15

U 1987, 37CE,15  
U 1988, 3E8A,15

ZZ-ENKCF-1.4 : ENKCF.MIC Test 37 CSR DATA L15-MAR-83 K-1 Fiche 4 Frame K1 Sequence 628  
 : ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40 Page 622  
 : ENKCF.MIC Test 37 CSR DATA LATE (DLT) BITS TEST (M8388 IDC MODULE)

```

U 1989, 3678,15 :31151 MOV LS[BIT28] TO WR[0] ;CLR CRDY (BIT 7)
 :31152 ;BIT 28 PREVENTS IDC TIMEOUTS
U 198A, 17A0,15 :31153 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
 :31154
 :31155 READ.CSR.T37.2:
U 198B, 9090,15 :31156 MISC [SET.PORT.I.0] ;SELECT THE IDC
U 198C, 9780,15 :31157 PORT.READ PORT.ADRS[CSR] ;SET UP TO READ THE CSR
U 198D, 3B32,15 :31158 MOV PORT TO LS[PORT0] ;READ THE DATA FROM CSR AND
 :31159 ;SAVE IT IN LS
U 198E, 3732,15 :31160 MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
U 198F, 2F82,95 :31161 CLR WR[1] ;SET UP EXPECTED DATA
U 1990, 0869,3C :31162 JSR [CHECK.RESULT] ;CHECK THE DATA
U 1991, 8998,94 :31163 JMP [LOOP.T37.2] ;LOOP HERE IF ERR & LOE IS SET
 :31164
U 1992, FF82,15 :31165 INC LS[ERROR.NUMBER] ;SET UP FOR ERROR 3
 :31166
 :31167
 :31168 ;SET UP THE CSR TO SET DLT
 :31169
U 1993, B70C,15 :31170 MOV LS[SETCSR.F6] TO WR[0] ;SET UP THE CSR
U 1994, B720,95 :31171 MOV LS[SETCSR.MAINT] TO WR[1]
U 1995, AEC2,15 :31172 BIS WR[1] TO WR[0]
U 1996, 17A0,15 :31173 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
 :31174 ;MAINT=1,CRDY=0,F<2:0>=110
 :31175 ;SET.DLT
 :31176 -----
 :31177 -----
 :31178
 :31179 *****
 :31180 * IDC *
 :31181 *****
 :31182 -----
 :31183 :018:
 :31184 :=000
 :31185 :DIAG.IDLE:
 :31186
 :31187 BEN0/F0,
 :31188 BEN1/F1,
 :31189 BEN2/F2,
 :31190 NAD/DIAG.IDLE
 :31191 -----
 :31192 :01E:
 :31193 :=110
 :31194
 :31195 BEN0/CRDY.L,
 :31196 BEN2/MAINT,
 :31197 NAD/TEST.FUNC6
 :31198 ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
 :31199 ;DEPENDENT ON CSR <CRDY> AND <MAINT>
 :31200 -----
 :31201 :077:
 :31202 :=1*1
 :31203 :TEST.F6:
 :31204
 :31205 ;DO NOT INCR DAR BETWEEN HERE AND
 :31206 ;DIAG.IDLE OR DIAG.WAIT
 :31207
 :31208
 :31209
 :31210
 :31211
 :31212
 :31213
 :31214
 :31215
 :31216
 :31217
 :31218
 :31219
 :31220
 :31221
 :31222
 :31223
 :31224
 :31225
 :31226
 :31227
 :31228
 :31229
 :31230
 :31231
 :31232
 :31233
 :31234
 :31235
 :31236
 :31237
 :31238
 :31239
 :31240
 :31241
 :31242
 :31243
 :31244
 :31245
 :31246
 :31247
 :31248
 :31249
 :31250
 :31251
 :31252
 :31253
 :31254
 :31255
 :31256
 :31257
 :31258
 :31259
 :31260
 :31261
 :31262
 :31263
 :31264
 :31265
 :31266
 :31267
 :31268
 :31269
 :31270
 :31271
 :31272
 :31273
 :31274
 :31275
 :31276
 :31277
 :31278
 :31279
 :31280
 :31281
 :31282
 :31283
 :31284
 :31285
 :31286
 :31287
 :31288
 :31289
 :31290
 :31291
 :31292
 :31293
 :31294
 :31295
 :31296
 :31297
 :31298
 :31299
 :31300
 :31301
 :31302
 :31303
 :31304
 :31305
 :31306
 :31307
 :31308
 :31309
 :31310
 :31311
 :31312
 :31313
 :31314
 :31315
 :31316
 :31317
 :31318
 :31319
 :31320
 :31321
 :31322
 :31323
 :31324
 :31325
 :31326
 :31327
 :31328
 :31329
 :31330
 :31331
 :31332
 :31333
 :31334
 :31335
 :31336
 :31337
 :31338
 :31339
 :31340
 :31341
 :31342
 :31343
 :31344
 :31345
 :31346
 :31347
 :31348
 :31349
 :31350
 :31351
 :31352
 :31353
 :31354
 :31355
 :31356
 :31357
 :31358
 :31359
 :31360
 :31361
 :31362
 :31363
 :31364
 :31365
 :31366
 :31367
 :31368
 :31369
 :31370
 :31371
 :31372
 :31373
 :31374
 :31375
 :31376
 :31377
 :31378
 :31379
 :31380
 :31381
 :31382
 :31383
 :31384
 :31385
 :31386
 :31387
 :31388
 :31389
 :31390
 :31391
 :31392
 :31393
 :31394
 :31395
 :31396
 :31397
 :31398
 :31399
 :31400
 :31401
 :31402
 :31403
 :31404
 :31405
 :31406
 :31407
 :31408
 :31409
 :31410
 :31411
 :31412
 :31413
 :31414
 :31415
 :31416
 :31417
 :31418
 :31419
 :31420
 :31421
 :31422
 :31423
 :31424
 :31425
 :31426
 :31427
 :31428
 :31429
 :31430
 :31431
 :31432
 :31433
 :31434
 :31435
 :31436
 :31437
 :31438
 :31439
 :31440
 :31441
 :31442
 :31443
 :31444
 :31445
 :31446
 :31447
 :31448
 :31449
 :31450
 :31451
 :31452
 :31453
 :31454
 :31455
 :31456
 :31457
 :31458
 :31459
 :31460
 :31461
 :31462
 :31463
 :31464
 :31465
 :31466
 :31467
 :31468
 :31469
 :31470
 :31471
 :31472
 :31473
 :31474
 :31475
 :31476
 :31477
 :31478
 :31479
 :31480
 :31481
 :31482
 :31483
 :31484
 :31485
 :31486
 :31487
 :31488
 :31489
 :31490
 :31491
 :31492
 :31493
 :31494
 :31495
 :31496
 :31497
 :31498
 :31499
 :31500
 :31501
 :31502
 :31503
 :31504
 :31505
 :31506
 :31507
 :31508
 :31509
 :31510
 :31511
 :31512
 :31513
 :31514
 :31515
 :31516
 :31517
 :31518
 :31519
 :31520
 :31521
 :31522
 :31523
 :31524
 :31525
 :31526
 :31527
 :31528
 :31529
 :31530
 :31531
 :31532
 :31533
 :31534
 :31535
 :31536
 :31537
 :31538
 :31539
 :31540
 :31541
 :31542
 :31543
 :31544
 :31545
 :31546
 :31547
 :31548
 :31549
 :31550
 :31551
 :31552
 :31553
 :31554
 :31555
 :31556
 :31557
 :31558
 :31559
 :31560
 :31561
 :31562
 :31563
 :31564
 :31565
 :31566
 :31567
 :31568
 :31569
 :31570
 :31571
 :31572
 :31573
 :31574
 :31575
 :31576
 :31577
 :31578
 :31579
 :31580
 :31581
 :31582
 :31583
 :31584
 :31585
 :31586
 :31587
 :31588
 :31589
 :31590
 :31591
 :31592
 :31593
 :31594
 :31595
 :31596
 :31597
 :31598
 :31599
 :31600
 :31601
 :31602
 :31603
 :31604
 :31605
 :31606
 :31607
 :31608
 :31609
 :31610
 :31611
 :31612
 :31613
 :31614
 :31615
 :31616
 :31617
 :31618
 :31619
 :31620
 :31621
 :31622
 :31623
 :31624
 :31625
 :31626
 :31627
 :31628
 :31629
 :31630
 :31631
 :31632
 :31633
 :31634
 :31635
 :31636
 :31637
 :31638
 :31639
 :31640
 :31641
 :31642
 :31643
 :31644
 :31645
 :31646
 :31647
 :31648
 :31649
 :31650
 :31651
 :31652
 :31653
 :31654
 :31655
 :31656
 :31657
 :31658
 :31659
 :31660
 :31661
 :31662
 :31663
 :31664
 :31665
 :31666
 :31667
 :31668
 :31669
 :31670
 :31671
 :31672
 :31673
 :31674
 :31675
 :31676
 :31677
 :31678
 :31679
 :31680
 :31681
 :31682
 :31683
 :31684
 :31685
 :31686
 :31687
 :31688
 :31689
 :31690
 :31691
 :31692
 :31693
 :31694
 :31695
 :31696
 :31697
 :31698
 :31699
 :31700
 :31701
 :31702
 :31703
 :31704
 :31705
 :31706
 :31707
 :31708
 :31709
 :31710
 :31711
 :31712
 :31713
 :31714
 :31715
 :31716
 :31717
 :31718
 :31719
 :31720
 :31721
 :31722
 :31723
 :31724
 :31725
 :31726
 :31727
 :31728
 :31729
 :31730
 :31731
 :31732
 :31733
 :31734
 :31735
 :31736
 :31737
 :31738
 :31739
 :31740
 :31741
 :31742
 :31743
 :31744
 :31745
 :31746
 :31747
 :31748
 :31749
 :31750
 :31751
 :31752
 :31753
 :31754
 :31755
 :31756
 :31757
 :31758
 :31759
 :31760
 :31761
 :31762
 :31763
 :31764
 :31765
 :31766
 :31767
 :31768
 :31769
 :31770
 :31771
 :31772
 :31773
 :31774
 :31775
 :31776
 :31777
 :31778
 :31779
 :31780
 :31781
 :31782
 :31783
 :31784
 :31785
 :31786
 :31787
 :31788
 :31789
 :31790
 :31791
 :31792
 :31793
 :31794
 :31795
 :31796
 :31797
 :31798
 :31799
 :31800
 :31801
 :31802
 :31803
 :31804
 :31805
 :31806
 :31807
 :31808
 :31809
 :31810
 :31811
 :31812
 :31813
 :31814
 :31815
 :31816
 :31817
 :31818
 :31819
 :31820
 :31821
 :31822
 :31823
 :31824
 :31825
 :31826
 :31827
 :31828
 :31829
 :31830
 :31831
 :31832
 :31833
 :31834
 :31835
 :31836
 :31837
 :31838
 :31839
 :31840
 :31841
 :31842
 :31843
 :31844
 :31845
 :31846
 :31847
 :31848
 :31849
 :31850
 :31851
 :31852
 :31853
 :31854
 :31855
 :31856
 :31857
 :31858
 :31859
 :31860
 :31861
 :31862
 :31863
 :31864
 :31865
 :31866
 :31867
 :31868
 :31869
 :31870
 :31871
 :31872
 :31873
 :31874
 :31875
 :31876
 :31877
 :31878
 :31879
 :31880
 :31881
 :31882
 :31883
 :31884
 :31885
 :31886
 :31887
 :31888
 :31889
 :31890
 :31891
 :31892
 :31893
 :31894
 :31895
 :31896
 :31897
 :31898
 :31899
 :31900
 :31901
 :31902
 :31903
 :31904
 :31905
 :31906
 :31907
 :31908
 :31909
 :31910
 :31911
 :31912
 :31913
 :31914
 :31915
 :31916
 :31917
 :31918
 :31919
 :31920
 :31921
 :31922
 :31923
 :31924
 :31925
 :31926
 :31927
 :31928
 :31929
 :31930
 :31931
 :31932
 :31933
 :31934
 :31935
 :31936
 :31937
 :31938
 :31939
 :31940
 :31941
 :31942
 :31943
 :31944
 :31945
 :31946
 :31947
 :31948
 :31949
 :31950
 :31951
 :31952
 :31953
 :31954
 :31955
 :31956
 :31957
 :31958
 :31959
 :31960
 :31961
 :31962
 :31963
 :31964
 :31965
 :31966
 :31967
 :31968
 :31969
 :31970
 :31971
 :31972
 :31973
 :31974
 :31975
 :31976
 :31977
 :31978
 :31979
 :31980
 :31981
 :31982
 :31983
 :31984
 :31985
 :31986
 :31987
 :31988
 :31989
 :31990
 :31991
 :31992
 :31993
 :31994
 :31995
 :31996
 :31997
 :31998
 :31999
 :32000
 :32001
 :32002
 :32003
 :32004
 :32005
 :32006
 :32007
 :32008
 :32009
 :32010
 :32011
 :32012
 :32013
 :32014
 :32015
 :32016
 :32017
 :32018
 :32019
 :32020
 :32021
 :32022
 :32023
 :32024
 :32025
 :32026
 :32027
 :32028
 :32029
 :32030
 :32031
 :32032
 :32033
 :32034
 :32035
 :32036
 :32037
 :32038
 :32039
 :32040
 :32041
 :32042
 :32043
 :32044
 :32045
 :32046
 :32047
 :32048
 :32049
 :32050
 :32051
 :32052
 :32053
 :32054
 :32055
 :32056
 :32057
 :32058
 :32059
 :32060
 :32061
 :32062
 :32063
 :32064
 :32065
 :32066
 :32067
 :32068
 :32069
 :32070
 :32071
 :32072
 :32073
 :32074
 :32075
 :32076
 :32077
 :32078
 :32079
 :32080
 :32081
 :32082
 :32083
 :32084
 :32085
 :32086
 :32087
 :32088
 :32089
 :32090
 :32091
 :32092
 :32093
 :32094
 :32095
 :32096
 :32097
 :32098
 :32099
 :32100
 :32101
 :32102
 :32103
 :32104
 :32105
 :32106
 :32107
 :32108
 :32109
 :32110
 :32111
 :32112
 :32113
 :32114
 :32115
 :32116
 :32117
 :32118
 :32119
 :32120
 :32121
 :32122
 :32123
 :32124
 :32125
 :32126
 :32127
 :32128
 :32129
 :32130
 :32131
 :32132
 :32133
 :32134
 :32135
 :32136
 :32137
 :32138
 :32139
 :32140
 :32141
 :32142
 :32143
 :32144
 :32145
 :32146
 :32147
 :32148
 :32149
 :32150
 :32151
 :32152
 :32153
 :32154
 :32155
 :32156
 :32157
 :32158
 :32159
 :32160
 :32161
 :32162
 :32163
 :32164
 :32165
 :32166
 :32167
 :32168
 :32169
 :32170
 :32171
 :32172
 :32173
 :32174
 :32175
 :32176
 :32177
 :32178
 :32179
 :32180
 :32181
 :32182
 :32183
 :32184
 :32185
 :32186
 :32187
 :32188
 :32189
 :32190
 :32191
 :32192
 :32193
 :32194
 :32195
 :32196
 :32197
 :32198
 :32199
 :32200
 :32201
 :32202
 :32203
 :32204
 :32205
 :32206
 :32207
 :32208
 :32209
 :32210
 :32211
 :32212
 :32213
 :32214
 :32215
 :32216
 :32217
 :32218
 :32219
 :32220
 :32221
 :32222
 :32223
 :32224
 :32225
 :32226
 :32227
 :32228
 :32229
 :32230
 :32231
 :32232
 :32233
 :32234
 :32235
 :32236
 :32237
 :32238
 :32239
 :32240
 :32241
 :32242
 :32243
 :32244
 :32245
 :32246
 :32247
 :32248
 :32249
 :32250
 :32251
 :32252
 :32253
 :32254
 :32255
 :32256
 :32257
 :32258
 :32259
 :32260
 :32261
 :32262
 :32263
 :32264
 :32265
 :32266
 :32267
 :32268
 :32269
 :32270
 :32271
 :32272
 :32273
 :32274
 :32275
 :32276
 :32277
 :32278
 :32279
 :32280
 :32281
 :32282
 :32283
 :32284
 :32285
 :32286
 :32287
 :32288
 :32289
 :32290
 :32291
 :32292
 :32293
 :32294
 :32295
 :32296
 :32297
 :32298
 :32299
 :32300
 :32301
 :32302
 :32303
 :32304
 :32305
 :32306
 :32307
 :32308
 :32309
 :32310
 :32311
 :32312
 :32313
 :32314
 :32315
 :32316
 :32317
 :32318
 :32319
 :32320
 :32321
 :32322
 :32323
 :32324
 :32325
 :32326
 :32327
 :32328
 :32329
 :32330
 :32331
 :32332
 :32333
 :32334
 :32335
 :32336
 :32337
 :32338
 :32339
 :32340
 :32341
 :32342
 :32343
 :32344
 :32345
 :32346
 :32347
 :32348
 :32349
 :32350
 :32351
 :32352
 :32353
 :32354
 :32355
 :32356
 :32357
 :32358
 :32359
 :32360
 :32361
 :32362
 :32363
 :32364
 :32365
 :32366
 :32367
 :32368
 :32369
 :32370
 :32371
 :32372
 :32373
 :32374
 :32375
 :32376
 :32377
 :32378
 :32379
 :32380
 :32381
 :32382
 :32383
 :32384
 :32385
 :32386
 :32387
 :32388
 :32389
 :32390
 :32391
 :32392
 :32393
 :32394
 :32395
 :32396
 :32397
 :32398
 :32399
 :32400
 :32401
 :32402
 :32403
 :32404
 :32405
 :32406
 :32407
 :32408
 :32409
 :32410
 :32411
 :32412
 :32413
 :32414
 :32415
 :32416
 :32417
 :32418
 :32419
 :32420
 :32421
 :32422
 :32423
 :32424
 :32425
 :32426
 :32427
 :32428
 :32429
 :32430
 :32431
 :32432
 :32433
 :32434
 :32435
 :32436
 :32437
 :32438
 :32439
 :32440
 :32441
 :32442
 :32443
 :32444
 :32445
 :32446
 :32447
 :32448
 :32449
 :32450
 :32451
 :32452
 :32453
 :32454
 :32455
 :32456
 :32457
 :32458
 :32459
 :32460
 :32461
 :32462
 :32463
 :32464
 :32465
 :32466
 :32467
 :32468
 :32469
 :32470
 :32471
 :32472
 :32473
 :32474
 :32475
 :32476
 :32477
 :32478
 :32479
 :32480
 :32481
 :32482
 :32483
 :32484
 :32485
 :32486
 :32487
 :32488
 :32489
 :32490
 :32491
 :32492
 :32493
 :32494
 :32495
 :32496
 :32497
 :32498
 :32499
 :32500
 :32501
 :32502
 :32503
 :32504
 :32505
 :32506
 :32507
 :32508
 :32509
 :32510
 :32511
 :32512
 :32513
 :32514
 :32515
 :32516
 :32517
 :32518
 :32519
 :32520
 :32521
 :32522
 :32523
 :32524
 :32525
 :32526
 :32527
 :32528
 :32529
 :32530
 :32531
 :32532
 :32533
 :32534
 :32535
 :32536
 :32537
 :32538
 :32539
 :32540
 :32541
 :32542
 :32543
 :32544
 :32545
 :32546
 :32547
 :32548
 :32549
 :32550
 :32551
 :32552
 :32553
 :32554
 :32555
 :32556
 :32557
 :32558
 :32559
 :32560
 :32561
 :32562
 :32563
 :32564
 :32565
 :32566
 :32567
 :32568
 :32569
 :32570
 :32571
 :32572
 :32573
 :32574
 :32575
 :32576
 :32577
 :32578
 :32579
 :32580
 :32581
 :32582
 :32583
 :32584
 :32585
 :32586
 :32587
 :32588
 :32589
 :32590
 :32591
 :32592
 :32593
 :32594
 :32595
 :3
```

ZZ-ENKCF-1.4 :  
: ENKCF.MCR  
: ENKCF.MIC

ENKCF.MIC

Test 37 CSR DATA L15-MAR-83  
MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module  
Test 37 CSR DATA LATE (DLT) BITS TEST (M8388 IDC MODULE)

Fiche 4 Frame L1

Sequence 629  
Rev 01.40 Page 623

ZZ-E  
:  
:

```
:31206 : UCMD/SET.DLT, ;SET DLT FOR DLT TEST
:31207 :
:31208 : FUNC/SET.INT, ;SET INT FOR INTERRUPT TEST
:31209 :
:31210 : NAD/XFER.REQ ;SEND XFER REQ TO FLAG THE CPU
:31211 : -----
:31212 : 1D8:
:31213 : XFER.REQ:
:31214 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
:31215 : NAD/DIAG.WAIT
:31216 : -----
:31217 : 1D7:
:31218 : DIAG.WAIT:
:31219 :
:31220 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:31221 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:31222 : -----
:31223 : 159:
:31224 : =*0*
:31225 : DIAG.WAIT1:
:31226 :
:31227 : NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:31228 : ; COMMAND FROM THE CPU
:31229 : -----
:31230 : 15B:
:31231 : =*1*
:31232 : DIAG.WAIT2:
:31233 :
:31234 : NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....
:31235 : ; WAIT SOME MORE
:31236 : -----
:31237 : WAIT.IDC.T37.3:
U 1997, DB00,1A :31238 : SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ FROM IDC
U 1998, 0999,A4 :31239 : JMP [IDC.DONE.T37.3] ;XFER REQ SET
U 1999, 8999,74 :31240 : JMP [WAIT.IDC.T37.3] ;WAIT SOME MORE
:31241 :
:31242 : IDC.DONE.T37.3:
U 199A, B700,15 :31243 : MOV LS[SETCSR.F0] TO WR[0] ;CLEAR CSR F<2:0>
U 199B, 3712,95 :31244 : MOV LS[SETCSR.CRDY] TO WR[1] ;SET CRDY AS NOT TO CLR DLT
U 199C, AEC2,15 :31245 : BIS WR[1] TO WR[0]
U 199D, 17A0,15 :31246 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
U 199E, 90FA,15 :31247 : MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
:31248 :
:31249 : LOOP.T37.3:
U 199F, 17CC,15 :31250 : PORT.CONTROL [CLEAR.IDC] ;INIT THE IDC
U 19A0, 17CC,15 :31251 : PORT.CONTROL [CLEAR.IDC]
U 19A1, 17CC,15 :31252 : PORT.CONTROL [CLEAR.IDC]
:31253 :
:31254 : READ.CSR.T37.3:
U 19A2, 9090,15 :31255 : MISC [SET.PORT.I.0] ;SELECT THE IDC
U 19A3, 9780,15 :31256 : PORT.READ PORT.ADRS[CSR] ;SET UP TO READ THE CSR
U 19A4, 3B32,15 :31257 : MOV PORT TO LS[PORT0] ;READ THE DATA FROM CSR AND
:31258 : ; SAVE IT IN LS
U 19A5, 3732,15 :31259 : MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
U 19A6, 2F82,95 :31260 : CLR WR[1] ;SET UP EXPECTED DATA
```

U 1  
U 1  
U 1  
U 1  
U 1  
U 1  
U 1  
U 1  
U 1



:31270 .PAGE "Test 38 SKIP SECTOR ERROR FLAG TEST (M8388 IDC MODULE)"

:31271 :++

```

:31272 :
:31273 : Test Description:

```

```

:31274 :
:31275 : This test will verify the Skip Sector Error Flag, CSR bit <23> and
:31276 : INH SSE, CSR bit <22> interact correctly.
:31277 : Both bits will first be cleared by writing to the CSR.
:31278 : Then a USET.SSE will be executed from the IDC microcode. The CPU
:31279 : will then read the CSR and check that bit <23> is set and bit <22>
:31280 : is clear. Bit <23> will then be cleared by writing to the CSR.
:31281 : A USET.SSE will then be executed with CSR <22> (INH SSE) set and
:31282 : the CPU will read and check that bit <23> (SSE FLAG) did not get
:31283 : latched up and that bit <22> remained set.

```

```

:31284
:31285 : Assumptions:

```

```

:31286 :
:31287 : It is assumed that the CSR test has run successfully.

```

```

;31289 ; Test Steps:

```

- ```

:31290 :
:31291 : 1. Set up error mask, error number and module number in LS
:31292 : (for error printouts) and clear the previous error number in LS.
:31293 : 2. Force the IDC microcode to DIAG.IDLE.
:31294 : 3. Clear CSR <23:22>.
:31295 : 4. Set up the CSR to branch to the correct IDC microcode routine.

```

```

:31298 : *****
:31299 :  * IDC *
:31300 : *****

```

```

:31301 :
:31302 : IDC1. Execute an instruction with UFUNC <3:0> = 1000,
:31303 : USET.SSE.

```

```

:31303 :
:31304 :      IDC2. Issue an XFER.REQ to flag the CPU and go to DIAG.WAIT.

```

- ```

:31305 :
:31306 : 5. Wait for a XFER REQ from IDC.
:31307 : 6. Clear CSR F<2:0> and send XFER GRANT to clear XFER.REQ.
:31308 : 7. Check CSR <23> = 1 and CSR <22> = 0.
:31309 : 8. Increment the error number.
:31310 : 9. Clear CSR <23>.
:31311 : 10. Set CSR <22>, INH SSE.
:31312 : 11. Set up the CSR to branch to the correct microcode routine.

```

```

:31315 : *****
:31316 : * IDC *
:31317 : *****

```

31318 :  
31319 : IDC3. Perform Steps IDC1 - IDC2.

- ```

:31320 :
:31321 : 12. Wait for a XFER REQ from IDC.
:31322 : 13. Clear CSR F<2:0> and send XFER GRANT to clear the XFER REQ.
:31323 : 14. Check CSR <23> = 0 and CSR <22> = 1.
:31324 : 15. Clear IDC.

```

[illegible]


```

31325 : 16. End test.
31326 :
31327 : Errors:
31328 :
31329 :      Printout:
31330 :      EXPECTED      RECEIVED
31331 :      CSR           CSR
31332 :
31333 :
31334 :      ERROR 1 - When a USET.SSE was done from the IDC microcode, either
31335 :      Bit <23> did not set or bit <22> did not remain clear.
31336 :
31337 :      ERROR 2 - Bit <23> was latched by a USET.SSE when INH SSE
31338 :      <22> was set.
31339 :
31340 :
31341 : Debug:
31342 :
31343 : ERROR 1 -
31344 : First CSR <23:22> are written to zeros.
31345 : Then the IDC branches to a microinstruction with UFUNC<3:0> = 1000.
31346 : These bits are decoded and will assert USET SSE L. This is an input
31347 : to the ONLINE REG PAL and at the next P2 CLOCK L, SSE FLAG L will
31348 : assert. SSE FLAG L will remain latched as long as INH SSE L is not
31349 : asserted. SSE FLAG L becomes bit 23 of the CSR.
31350 :
31351 : ERROR 2 -
31352 : The CPU issues a PORT.WRITE PORT.ADRS[CSR] with bits <23:22> = 01.
31353 : This instruction is decoded by the PORT RD/WR DECODE PAL and asserts
31354 : WRITE CSR L. The inputs to the ONLINE REG PAL should be:
31355 :      WRITE CSR L = L
31356 :      BUS IO 23 H = L
31357 :      BUS IO 22 H = H
31358 : This will assert INH SSE L. The IDC will then branch to an instruction
31359 : that will assert USET SSE L (see ERROR 1), however since INH SSE L is
31360 : asserted, SSE FLAG L will not be latched.
31361 : --
31362 : *****
31363 : T.38:
31364 :      MOV LS[BEGIN.TEST] TO WR[0]      ;SET BIT 15 IN WR[0] FOR
31365 :      MOV WR[0] TO LS[CONTROL.STATUS]  ;CONTROL AND STATUS WORD
31366 :      ;SET BIT 15 IN CONTROL AND
31367 :      ;STATUS WORD - BIT 15
31368 :      ;INDICATES START OF TEST TO
31369 :      ;CONSOLE
31370 :      MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
31371 : WAIT.T38.0:
31372 :      JMP [WAIT.T38.0]                ;LOOP HERE WAITING FOR
31373 :      ;CONSOLE TO RESPOND
31374 :
31375 :      CLR LS[PREVIOUS.ERROR]          ;CLEAR PREVIOUS ERROR NUMBER
31376 :      ;IN LS
31377 :
31378 :      MOV LS[IDC] TO WR[0]            ;STORE MODULE CODE IN LS TO
31379 :      MOV WR[0] TO LS[MODULE.NUM]     ;INDICATE IDC MODULE
  
```

U 19AA, B65E,15

U 19AB, 3E80,15

U 19AC, 10E0,15

U 19AD, 899A,D4

U 19AE, 65A0,15

U 19AF, B64A,15

U 19B0, 3E8C,15

ZZ-

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

[illegible]

```

31435 ; FUNCTION BITS
31436 -----
31437 :186:
31438 : =1**
31439
31440 BEN0/F0,
31441 BEN1/F1, ; BRANCH ON THE CSR F<2:0> BITS
31442 BEN2/F2,
31443 NAD/GROUP1.F0
31444 -----
31445 :02F:
31446 : =111
31447
31448 UCMD/SET.OPI, ; SET OPI FOR CSR OPI TEST
31449
31450 FUNC/SET.SSE, ; SET SKIP SECTOR ERROR
31451 NAD/OPI.BR.TST ; USED FOR OPI.L BRANCH TEST...DOES NOT AFFECT
31452 ; SET SSE OR SET.OPI
31453 -----
31454 :02B:
31455 : =011
31456 OPI.BR.TST:
31457
31458 BEN2/OPI.L, ; IS OPI ASSERTED?
31459 NAD/BEN2.1
31460 -----
31461 :1A0:
31462 : =0**
31463 BEN2.1:
31464
31465 INCR.DAR/ON, ; INCR DAR
31466 NAD/XFER.REQ ; FLAG THE CPU
31467 -----
31468 :1A4:
31469 : =1**
31470 BEN2.2:
31471
31472 INCR.DAR/ON, ; INCR DAR IF BEN2 CONDITION ASSERTED
31473 NAD/BEN2.1 ; INCR DAR AGAIN IF BEN2 COND ASSERTED
31474 -----
31475 :1D8:
31476 XFER.REQ:
31477 UCMD/SET.XFER.RQST, ; SEND XFER REQ TO THE CPU
31478 NAD/DIAG.WAIT
31479 -----
31480 :1D7:
31481 DIAG.WAIT:
31482
31483 BEN1/XFER.REQ, ; WAIT FOR THE CPU TO ISSUE XFER GRANT
31484 NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
31485 -----
31486 :159:
31487 : =*0*
31488 DIAG.WAIT1:
31489

```

U 1

U 1

U 1

U 1

101

U 1

10 1
4 1

1011

U i

01

```

31545 :DIAG.IDLE:
31546 :
31547 :      BEN0/F0,
31548 :      BEN1/F1,                ;BRANCH ON THE CSR F<2:0> BITS
31549 :      BEN2/F2,
31550 :      NAD/DIAG.IDLE
31551 :-----
31552 :01F:
31553 :=111
31554 :
31555 :      BEN0/CRDY.L,
31556 :      BEN2/MAINT,            ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
31557 :      NAD/TEST.FUNC7        ; DEPENDENT ON CSR <CRDY> AND <MAINT>
31558 :-----
31559 :079:
31560 :=0*1
31561 :
31562 :      NAD/GROUP1
31563 :-----
31564 :182:
31565 :=0**
31566 :GROUP1:
31567 :
31568 :      BEN2/R80.FORMAT.H,      ;WHEN THE R80 FORMAT BIT SETS..BRANCH
31569 :      NAD/GROUP1             ; TO NEXT WORD AND THEN BRANCH ON THE
31570 :                               ; FUNCTION BITS
31571 :-----
31572 :186:
31573 :=1**
31574 :
31575 :      BEN0/F0,
31576 :      BEN1/F1,                ;BRANCH ON THE CSR F<2:0> BITS
31577 :      BEN2/F2,
31578 :      NAD/GROUP1.F0
31579 :-----
31580 :02F:
31581 :=111
31582 :
31583 :      UCMD/SET.OPI,          ;SET OPI FOR CSR OPI TEST
31584 :
31585 :      FUNC/SET.SSE,          ;SET SKIP SECTOR ERROR
31586 :      NAD/OPI.BR.TST        ;USED FOR OPI.L BRANCH TEST...DOES NOT AFFECT
31587 :                               ; SET SSE OR SET.OPI
31588 :-----
31589 :02B:
31590 :=011
31591 :OPI.BR.TST:
31592 :
31593 :      BEN2/OPI.L,            ;IS OPI ASSERTED?
31594 :      NAD/BEN2.1
31595 :-----
31596 :1A0:
31597 :=0**
31598 :BEN2.1:
31599 :

```

[illegible]

ZZ-ENKCF-1.4 :
: ENKCF.MCR
: ENKCF.MIC

ENKCF.MIC

MICRO2 1M(01)

Test 38 SKIP SECT015-MAR-83
15-MAR-83 11:46:22
Test 38 SKIP SECTOR ERROR FLAG TEST (M8388 IDC MODULE)

Fiche 4 Frame G2
ENKCF Micro-diagnostic for IDC module

Sequence 637
Rev 01.40 Page 631

ZZ-1
:
:

U 1/
U 1/
U 1/
U 1/
U 1/
U 1/
U 1/

```
31600 : INCR.DAR/ON, ; INCR DAR
31601 : NAD/XFER.REQ ; FLAG THE CPU
31602 : -----
31603 : 1A4:
31604 : =1**
31605 : BEN2.2:
31606 :
31607 : INCR.DAR/ON, ; INCR DAR IF BEN2 CONDITION ASSERTED
31608 : NAD/BEN2.1 ; INCR DAR AGAIN IF BEN2 COND ASSERTED
31609 : -----
31610 : 1D8:
31611 : XFER.REQ:
31612 : UCMD/SET.XFER.RQST, ; SEND XFER REQ TO THE CPU
31613 : NAD/DIAG.WAIT
31614 : -----
31615 : 1D7:
31616 : DIAG.WAIT:
31617 :
31618 : BEN1/XFER.REQ, ; WAIT FOR THE CPU TO ISSUE XFER GRANT
31619 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
31620 : -----
31621 : 159:
31622 : =*0*
31623 : DIAG.WAIT1:
31624 :
31625 : NAD/DIAG.IDLE ; RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
31626 : ; COMMAND FROM THE CPU
31627 : -----
31628 : 15B:
31629 : =*1*
31630 : DIAG.WAIT2:
31631 :
31632 : NAD/DIAG.WAIT ; XFER GRANT HAS NOT BEEN ISSUED....
31633 : ; WAIT SOME MORE
31634 : -----
31635 : WAIT.IDC.T38.2:
U 19D1, DB00,1A 31636 : SKIP.IF[NO.FAST.INTERRUPT] ; WAIT FOR XFER REQ FROM IDC
U 19D2, 099D,44 31637 : JMP [IDC.DONE.T38.2] ; XFER REQ SET
U 19D3, 099D,14 31638 : JMP [WAIT.IDC.T38.2] ; WAIT SOME MORE
31639 :
31640 : IDC.DONE.T38.2:
U 19D4, 9090,15 31641 : MISC [SET.PORT.I.0] ; SELECT THE IDC
U 19D5, 9780,15 31642 : PORT.READ PORT.ADRS[CSR] ; SET UP TO READ THE CSR
U 19D6, 3B32,15 31643 : MOV PORT TO LS[PORT0] ; READ THE DATA FROM CSR AND
31644 : ; SAVE IT IN LS
31645 : ; SEND XFER GRANT TO CLEAR THE XFER REQ
U 19D7, B700,15 31646 : MOV LS[SETCSR.F0] TO WR[0] ; CLEAR CSR F<2:0>
U 19D8, 17A0,15 31647 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ; WRITE THE CSR
U 19D9, 90FA,15 31648 : MISC2 [TRANSFER.GRANT] ; SEND GRANT TO CLR THE XFER REQ
31649 :
U 19DA, 3732,15 31650 : MOV LS[PORT0] TO WR[0] ; SET UP RECEIVED DATA
U 19DB, B66C,95 31651 : MOV LS[BIT22] TO WR[1] ; SET UP EXPECTED DATA
U 19DC, 0869,3C 31652 : JSR [CHECK.RESULT] ; CHECK THE DATA
U 19DD, 099C,A4 31653 : JMP [LOOP.T38.2] ; LOOP HERE IF ERR AND LOE SET
31654 :
```

[illegible]

31662 .PAGE 'Test 39 INTERRUPT REQUEST TEST 1 (M8388 IDC MODULE)'

31663 :++

31664 :
31665 : Test Description:

31666 :
31667 : This test is designed to insure that the IDC interrupt request at BR 5
31668 : logic is working correctly. This will be accomplished by executing an
31669 : IDC microinstruction with FUNC<35:32> = 0C, SET.INT with CSR <INT.IE>
31670 : set and CRDY set. The CPU will check the interrupt line to be sure that
31671 : the interrupt was issued. The CPU will then issue a RESET BR Port
31672 : Instruction and will check that the interrupt request was cleared.
31673 : SET.INT will then be issued with CRDY clear and CSR <INT.IE> set.
31674 : The CPU will check that the interrupt was not received.
31675 : CSR <INT.IE> will then be cleared and CRDY will be set. An IDC
31676 : microinstruction with FUNC<35:32> = 0C, SET.INT will then be executed.
31677 : The CPU will check to insure that the interrupt did not occur.

31678 :
31679 : Assumptions:

31680 :
31681 : It is assumed that all previous tests have run successfully.

31682 :
31683 : Test Steps:

- 31684 :
31685 : 1. Set up error mask, error number and module number in LS
31686 : (for error printouts) and clear the previous error number in LS.
31687 : 2. Force IDC microcode to DIAG.IDLE.
31688 : 3. Lower the IPL to 0. (CLR LS[PSL.HW])
31689 : 4. Set bit <INTERUPT.EN> of the LS[CONTROL.STATUS] and allow time
31690 : any pending interrupts to clear.
31691 : 5. Set IDC CSR <INT.IE>, interrupt enable and set CSR CRDY.
31692 : 6. Set up the CSR to branch to the correct IDC microcode routine.

31693 :
31694 :
31695 : *****
31696 : * IDC *
31697 : *****

31698 :
31699 : IDC1. Execute an instruction with FUNC <35:32> = 0C, SET.INT.
31700 : IDC2. Issue XFER.REQ to flag the CPU.

- 31701 :
31702 : 7. Wait for XFER.REQ from the IDC.
31703 : 8. Clear CSR F<2:0> and send XFER GRANT to clear the XFER.REQ.
31704 : 9. If the hardware register (Interrupt Vector does not decode a BR5
31705 : logic code, then issue error 6 (this is an added error report for
31706 : fault insertion enhancement).
31707 : 10. Check the IR bit of CSR, it should be set, if not issue error 7
31708 : (this is an added error report for fault insertion enhancement).
31709 : 11. If Interrupt Request is not set then issue ERROR 1.
31710 : 12. Increment the error number to ERROR 2.
31711 : 13. Issue port instruction to clear the BR 5.
31712 : 14. Check the IR bit of CSR, it should be reset, if not issue error 8
31713 : (this is an added error report for fault insertion enhancement).
31714 : 15. If Interrupt Request is still asserted then issue ERROR.
31715 : 16. Increment the error number.
31716 : 17. Clear CSR CRDY and set CSR <INT.IE>.

:31717 : 18. Set up the CSR to branch to the correct IDC microcode routine.
:31718 :
:31719 :
:31720 : *****
:31721 : * IDC *
:31722 : *****
:31723 :
:31724 : IDC3. Execute an instruction with FUNC <35:32> = 0C, SET.INT.
:31725 : IDC4. Issue XFER.REQ to flag the CPU.
:31726 :
:31727 : 19. Wait for XFER.REQ from the IDC.
:31728 : 20. Clear CSR F<2:0> and send XFER GRANT to clear the XFER.REQ.
:31729 : 21. If Interrupt Request is asserted then issue error.
:31730 : 22. Increment the error number.
:31731 : 23. Clear CSR <INT.IE> and set CSR CRDY.
:31732 : 24. Set up the CSR to branch to the correct IDC microcode routine.
:31733 :
:31734 :
:31735 :
:31736 :
:31737 :
:31738 :
:31739 : *****
:31740 : * IDC *
:31741 : *****
:31742 :
:31743 : IDC5. Execute an instruction with FUNC <35:32> = 0C, SET.INT.
:31744 : IDC6. Issue XFER.REQ to flag the CPU.
:31745 :
:31746 : 25. Clear CSR F<2:0> and send XFER GRANT to clear XFER.REQ.
:31747 : 26. If Interrupt Request is asserted then issue ERROR.
:31748 : 27. Increment the error number.
:31749 : 28. Set Interrupt Request.
:31750 : 29. INIT the IDC.
:31751 : 30. If Interrupt Request is asserted then issue error.
:31752 : 31. Clear IDC.
:31753 : 32. End test.
:31754 :
:31755 : Errors:
:31756 :
:31757 : ERROR 1 - IDC did not assert the interrupt request.
:31758 :
:31759 : ERROR 2 - Interrupt Request did not clear when RESET BR was issued.
:31760 :
:31761 : ERROR 3 - Interrupt Request asserted when CRDY was cleared.
:31762 :
:31763 : ERROR 4 - Interrupt Request asserted when not enabled.
:31764 :
:31765 : ERROR 5 - Interrupt Request was asserted after IDC INIT.
:31766 :
:31767 : ERROR 6 - BR5 Interrupt was not seen by cpu after SET.INT was asserted.
:31768 :
:31769 : ERROR 7 - IR bit of CSR was not set after SET.INT was asserted.
:31770 :
:31771 : ERROR 8 - IR was not cleared after clearing the Interrupt Request.

MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module
Test 39 INTERRUPT REQUEST TEST 1 (M8388 IDC MODULE)

Rev 01.40

Page 635

```

31772 : Debug:
31773 :
31774 :
31775 : ERROR 1-
31776 : Any pending interrupts are cleared and the IPL is lowered to 0.
31777 : CSR <CRDY> and CSR <IE> are set. The IDC microcode then branches to
31778 : a microinstruction with UFUNC<3:0>=1100 which is decoded and will
31779 : assert USET INT L. This is an input to the ATTN REG PAL.
31780 : USET INT L is gated through the PAL and will assert SET INT REQ H.
31781 : This signal is an input to the FIFO A ADRS CNTR LO PAL. The inputs at
31782 : this time should be:
31783 :     SET INT REQ H = H
31784 :     RESET BR L = H
31785 :     IE H = H
31786 :     CRDY H = H
31787 : This combination of inputs will assert INT REQ DIN H. This is an input
31788 : to the INIT FF and at the next P2 CLOCK L will assert INT REQ H.
31789 : INT REQ H is inverted and leaves the board as UBUS BR5 L.
31790 :
31791 : ERROR 2-
31792 : The CPU will issue a PORT.CONTROL [RESET.BR] to the IDC which
31793 : will clear the BR 5. The instruction is decoded by the PORT RD/WR
31794 : DECODE PAL. The inputs should be:
31795 :     CSR 17 H = H
31796 :     CSR 14 H = H
31797 :     CSR 12 H = L
31798 :     CSR 11 H = L
31799 :     CSR 10 H = H
31800 :     PORT INSTR H = H
31801 : This combination of inputs will assert RESET BR L. This is an input to
31802 : the FIFOA ADRS CNTR LO PAL and will clear INT REQ DIN H. This signal
31803 : is gated through the INIT FF at the next P2 CLOCK L forcing
31804 : INT REQ H = L. INT REQ H = L is inverted and leaved the board as
31805 : UBUS BR5 L = H.
31806 :
31807 : ERROR 3-
31808 : The IDC is set up to branch to a microinstruction with UFUNC<3:0>=1100
31809 : which is decoded and will assert USET INT L. This is an input to the
31810 : ATTN REG PAL. USET INT L is gated through the PAL and will assert
31811 : SET INT REQ H.
31812 : This signal is an input to the FIFO A ADRS CNTR LO PAL. The inputs at
31813 : this time should be:
31814 :     SET INT REQ H = H
31815 :     RESET BR L = H
31816 :     IE H = H
31817 :     CRDY H = L
31818 : Since CRDY is clear at this time, INT REQ DIN H will not assert.
31819 :
31820 : ERROR 4 -
31821 : The IDC is set up to branch to a microinstruction with UFUNC<3:0>=1100
31822 : which is decoded and will assert USET INT L. This is an input to the
31823 : ATTN REG PAL. USET INT L is gated through the PAL and will assert
31824 : SET INT REQ H.
31825 : This signal is an input to the FIFO A ADRS CNTR LO PAL. The inputs at
31826 : this time should be:

```

10 11

U 1.

U 14

10 14

U 1.

1014

01

101

64

101

1


```

U 19E9, 3E8C,15 :31882      MOV WR[0] TO LS[MODULE.NUM]      ; INDICATE IDC MODULE
                  :31883
U 19EA, 8872,EC :31884      JSR [DIAG.CODE]                ; INIT THE IDC AND FORCE IT TO
                  :31885                        ; DIAG.IDLE
                  :31886
U 19EB, 2F80,15 :31887      CLR WR[0]
U 19EC, BFFE,15 :31888      MOV WR[0] TO LS[PSL.HW]          ; LOWER THE IPL TO 0
                  :31889
                  :31890      LOOP.T39.1:
U 19ED, 370E,15 :31891      MOV LS[SETCSR.F7] TO WR[0]      ; SET UP THE CSR TO BRANCH TO
U 19EE, B720,95 :31892      MOV LS[SETCSR.MAINT] TO WR[1]    ; THE CORRECT IDC ROUTINE
U 19EF, AEC2,15 :31893      BIS WR[1] TO WR[0]
U 19F0, 3712,95 :31894      MOV LS[SETCSR.CRDY] TO WR[1]    ; SET CRDY
U 19F1, AEC2,15 :31895      BIS WR[1] TO WR[0]
U 19F2, B710,95 :31896      MOV LS[SETCSR.IE] TO WR[1]      ; SET INTERRUPT ENABLE
U 19F3, AEC2,15 :31897      BIS WR[1] TO WR[0]
U 19F4, 17A0,15 :31898      PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ; WRITE THE CSR
                  :31899                        ; MAINT=1,CRDY=1,F<2:0>=111
                  :31900      ; SET.INT
                  :31901      -----
                  :31902      -----
                  :31903      *****
                  :31904      * IDC *
                  :31905      *****
                  :31906      -----
                  :31907      :018:
                  :31908      :=000
                  :31909      :DIAG.IDLE:
                  :31910
                  :31911      BEN0/F0,
                  :31912      BEN1/F1,      ; BRANCH ON THE CSR F<2:0> BITS
                  :31913      BEN2/F2,
                  :31914      NAD/DIAG.IDLE
                  :31915      -----
                  :31916      :01F:
                  :31917      :=111
                  :31918
                  :31919      BEN0/CRDY.L,
                  :31920      BEN2/MAINT,    ; BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
                  :31921      NAD/TEST.FUNC7 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
                  :31922      -----
                  :31923      :07C:
                  :31924      :=1*0
                  :31925      :SET.INT:
                  :31926
                  :31927      FUNC/SET.INT,    ; SET INT FOR INT TEST...CRDY MUST
                  :31928                        ; BE SET IN CSR
                  :31929      NAD/XFER.REQ
                  :31930      -----
                  :31931      :1D8:
                  :31932      :XFER.REQ:
                  :31933      UCMD/SET.XFER.RQST, ; SEND XFER REQ TO THE CPU
                  :31934      NAD/DIAG.WAIT
                  :31935      -----
                  :31936      :1D7:
  
```

```

:31937 :DIAG.WAIT:
:31938
:31939 :      BEN1/XFER.REQ,      :WAIT FOR THE CPU TO ISSUE XFER GRANT
:31940 :      NAD/DIAG.WAIT1      : BEFORE RETURNING TO DIAG.IDLE
:31941 -----
:31942 :159:
:31943 :=*0*
:31944 :DIAG.WAIT1:
:31945
:31946 :      NAD/DIAG.IDLE      :RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:31947 :      : COMMAND FROM THE CPU
:31948 -----
:31949 :15B:
:31950 :=*1*
:31951 :DIAG.WAIT2:
:31952
:31953 :      NAD/DIAG.WAIT      :XFER GRANT HAS NOT BEEN ISSUED....
:31954 :      : WAIT SOME MORE
:31955 -----
:31956 :WAIT.IDC.T39.1:
:31957 :      SKIP.IF[NO.FAST.INTERRUPT]      :WAIT FOR XFER REQ FROM IDC
:31958 :      JMP [IDC.DONE.T39.1]      :XFER REQ SET
:31959 :      JMP [WAIT.IDC.T39.1]      :WAIT SOME MORE
:31960
:31961 :IDC.DONE.T39.1:
:31962 :      MOV LS[SETCSR.F0] TO WR[0]      :CLEAR CSR F<2:0>
:31963 :      MOV LS[SETCSR.CRDY] TO WR[1]      :SET CRDY
:31964 :      BIS WR[1] TO WR[0]
:31965 :      MOV LS[SETCSR.IE] TO WR[1]      :SET INT ENABLE
:31966 :      BIS WR[1] TO WR[0]
:31967 :      PORT.WRITE PORT.ADRS[CSR] WITH WR[0] :WRITE THE CSR
:31968 :      MISC2 [TRANSFER.GRANT]      :SEND GRANT TO CLR THE XFER REQ
:31969 :      JMP.IF[NO.INTERRUPT] TO [ERROR.T39.1] :JMP IF NO INTERRUPT (ERROR)
:31970 :      MOV LS[#6] TO WR[0]      :SET UP ERROR NUMBER = 6
:31971 :      MOV WR[0] TO LS[ERROR.NUMBER]
:31972 :      MOV LS[#FFFFFFE3] TO WR[0]      :MASK FOR BITS 4,3 & 2
:31973 :      MOV WR[0] TO LS[ERROR.MASK]
:31974 :      MOV LS[INTERRUPT.VEC] TO WR[0]      :FETCH THE INTER. HARDWARE REG.
:31975 :      MOV LS[#10] TO WR[1]      :EXP BR5 CODES SET IN H.D. REG.
:31976 :      JSR [CHECK.RESULT]      :CHECK IF BR5 LOGIC SEEN BY CPU
:31977 :      JMP [LOOP.T39.1]      :LOOP HERE IF LOOP IS SET
:31978
:31979 :      MISC [SET.PORT.I.0]      :SELECT FROM PORT TO BUS
:31980 :      PORT.READ PORT.ADRS[CSR]      :READ CSR BACK
:31981 :      MOV PORT TO LS[PORT0]      :SAVE IT FOR LATER
:31982 :      INC LS[ERROR.NUMBER]      :SET UP ERROR NUMBER = 7
:31983 :      MOV LS[#FFFFFFF] TO WR[0]      :MASK FOR BIT 24
:31984 :      MOV WR[0] TO LS[ERROR.MASK]
:31985 :      MOV LS[PORT0] TO WR[0]      :RECEIVED IR BIT FROM CSR
:31986 :      MOV LS[BIT24] TO WR[1]      :EXP. STATE OF CSR IR BIT
:31987 :      JSR [CHECK.RESULT]      :SEE IF IR BIT IS SET
:31988 :      JMP [LOOP.T39.1]      :LOOP HERE IF LOOP IS SET
:31989
:31990 :      MOV LS[#1] TO WR[0]
:31991 :      MOV WR[0] TO LS[ERROR.NUMBER]      :SET UP ERROR NUMBER = 1 IN LS
  
```

U 19F5, DB00,1A
 U 19F6, 899F,84
 U 19F7, 099F,54

U 19F8, B700,15
 U 19F9, 3712,95
 U 19FA, AEC2,15
 U 19FB, B710,95
 U 19FC, AEC2,15
 U 19FD, 17A0,15
 U 19FE, 90FA,15
 U 19FF, 89A1,C7
 U 1A00, 3792,15
 U 1A01, BE82,15
 U 1A02, 37EA,15
 U 1A03, 3E8A,15
 U 1A04, 36FC,15
 U 1A05, B648,95
 U 1A06, 0869,3C
 U 1A07, 099E,D4

U 1A08, 9090,15
 U 1A09, 9780,15
 U 1A0A, 3832,15
 U 1A0B, FF82,15
 U 1A0C, 37EC,15
 U 1A0D, 3E8A,15
 U 1A0E, 3732,15
 U 1A0F, 3670,95
 U 1A10, 0869,3C
 U 1A11, 099E,D4

U 1A12, B640,15
 U 1A13, BE82,15

U 1A
 U 1A
 U 1A

U 1A
 U 1A
 U 1A
 U 1A
 U 1A
 U 1A

U 1A

U 1A
 U 1A
 U 1A
 U 1A
 U 1A

U 1A

U 1A
 U 1A
 U 1A
 U 1A
 U 1A

U 1A
 U 1A

U 1A
 U 1A

U 1A

U 1A
 U 1A

nn nn n nn nnnnnnnnnn

C 3

ZZ-ENKCF-1.4 : ENKCF.MIC Test 39 INTERRUPT R15-MAR-83 Fiche 4 Frame C3 Sequence 646
 : ENKCF.MCR MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40 Page 640
 : ENKCF.MIC Test 39 INTERRUPT REQUEST TEST 1 (M8388 IDC MODULE)

```

U 1A38, 0869,3C :32047      JSR [CHECK.RESULT]          ;SEE IF IR BIT IS SET
U 1A39, 09A2,E4 :32048      JMP [RESET.OK.T39.1]        ;LOOP HERE IF LOOP IS SET
                  :32049
U 1A3A, B788,15 :32050      MOV LS[#3] TO WR[0]
U 1A3B, BE82,15 :32051      MOV WR[0] TO LS[ERROR.NUMBER]    ;ERROR 3
U 1A3C, B66E,15 :32052      MOV LS[NA.EXP.REC] TO WR[0]      ;EXP AND REC DATA ARE NOT
U 1A3D, 3E80,15 :32053      MOV WR[0] TO LS[CONTROL.STATUS]    ; APPLICABLE
                  :32054
U 1A3E, E58A,15 :32055      CLR LS[ERROR.MASK]              ;TO FORCE ERRORS
                  :32056
                  :32057 LOOP1.T39.3:
                  :32058
                  :32059 ;SET UP THE CSR TO BRANCH TO THE ROUTINE THAT WILL SET INTERRUPT WITH
                  :32060 ; CSR CRDY CLR
                  :32061
U 1A3F, B70C,15 :32062      MOV LS[SETCSR.F6] TO WR[0]        ;SET UP THE CSR
U 1A40, B720,95 :32063      MOV LS[SETCSR.MAINT] TO WR[1]
U 1A41, AEC2,15 :32064      BIS WR[1] TO WR[0]
U 1A42, B710,95 :32065      MOV LS[SETCSR.IE] TO WR[1]        ;SET INTERRUPT ENABLE
U 1A43, AEC2,15 :32066      BIS WR[1] TO WR[0]
U 1A44, 17A0,15 :32067      PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
                  :32068 ;MAINT=1,CRDY=0,F<2:0>=110
                  :32069 ;SET.INT WITH CSR<CRDY> =0
                  :32070 -----
                  :32071 -----
                  :32072 ;*****
                  :32073 ; * IDC *
                  :32074 ;*****
                  :32075 -----
                  :32076 ;018:
                  :32077 ;=000
                  :32078 ;DIAG.IDLE:
                  :32079
                  :32080 ;BEN0/F0,
                  :32081 ;BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
                  :32082 ;BEN2/F2,
                  :32083 ;NAD/DIAG.IDLE
                  :32084 -----
                  :32085 ;01E:
                  :32086 ;=110
                  :32087
                  :32088 ;BEN0/CRDY.L,
                  :32089 ;BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
                  :32090 ;NAD/TEST.FUNC6 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
                  :32091 -----
                  :32092 ;077:
                  :32093 ;=1*1
                  :32094 ;TEST.F6:
                  :32095 ;DO NOT INCR DAR BETWEEN HERE AND
                  :32096 ;DIAG.IDLE OR DIAG.WAIT
                  :32097
                  :32098 ;LOAD.LOOP.CNTR.[00], ;LOAD LOOP CNTR WITH 00
                  :32099 ;UCMD/SET.DLT, ;SET DLT FOR DLT TEST
                  :32100
                  :32101
  
```

ZZ-

...

...

[illegible]


```

:32157
U 1A5A, FF82,15 :32158      INC LS[ERROR.NUMBER]                ;ERROR 4
:32159
:32160      LOOP.T39.4:
:32161
U 1A5B, 97C4,15 :32162      PORT.CONTROL [RESET.BR]            ;CLEAR ANY INTERRUPT REQUEST
:32163
U 1A5C, B788,15 :32164      MOV LS[#3] TO WR[0]                ;WAIT FOR THE INTERRUPT TO CLR
:32165
:32166      WAIT.BR.T39.2:
:32167      DEC WR[0],                          ;WAIT SOME MORE
:32168      DT(LONG)&SET.ALU.CC
:32169      JMP.IF[NEQ] TO [WAIT.BR.T39.2]      ;WAIT SOME SOME
:32170
:32171      ;SET UP THE CSR TO BRANCH TO THE ROUTINE THAT WILL SET.INT WITH
:32172      ; IDC CSR INTERRUPT ENABLE NOT ASSERTED
:32173
U 1A5F, 370E,15 :32174      MOV LS[SETCSR.F7] TO WR[0]          ;SET UP THE CSR TO BRANCH TO
U 1A60, B720,95 :32175      MOV LS[SETCSR.MAINT] TO WR[1]        ; THE CORRECT IDC ROUTINE
U 1A61, AEC2,15 :32176      BIS WR[1] TO WR[0]
U 1A62, 3712,95 :32177      MOV LS[SETCSR.CRDY] TO WR[1]        ;SET CRDY
U 1A63, AEC2,15 :32178      BIS WR[1] TO WR[0]
U 1A64, 17A0,15 :32179      PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:32180      ;MAINT=1,CRDY=1,F<2:0>=111
:32181      ;SET.INT
:32182      -----
:32183      -----
:32184      :
:32185      :          *****
:32186      :          * IDC *
:32187      :          *****
:32188      :
:32189      :018:
:32190      :=000
:32191      :DIAG.IDLE:
:32192      :
:32193      :      BEN0/F0,
:32194      :      BEN1/F1,                ;BRANCH ON THE CSR F<2:0> BITS
:32195      :      BEN2/F2,
:32196      :      NAD/DIAG.IDLE
:32197      :-----
:32198      :01F:
:32199      :=111
:32200      :
:32201      :      BEN0/CRDY.L,
:32202      :      BEN2/MAINT,            ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:32203      :      NAD/TEST.FUNC7        ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:32204      :-----
:32205      :07C:
:32206      :=1*0
:32207      :SET.INT:
:32208      :
:32209      :      FUNC/SET.INT,          ;SET INT FOR INT TEST...CRDY MUST
:32210      :      NAD/XFER.REQ          ; BE SET IN CSR
:32211      :-----
  
```

ZZ-ENKCF-1.4 :
: ENKCF.MCR
: ENKCF.MIC

ENKCF.MIC

Test 39 INTERRUPT R15-MAR-83
MICRO2 1M(01) 15-MAR-83 11:46:22
Test 39 INTERRUPT REQUEST TEST 1 (M8388 IDC MODULE)

F 3
Fiche 4 Frame F3
ENKCF Micro-diagnostic for IDC module

Sequence 649
Rev 01.40 Page 643

ZZ-
:
:

```
:32212 :1D8:  
:32213 :XFER.REQ:  
:32214 :UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU  
:32215 :NAD/DIAG.WAIT  
:32216 -----  
:32217 :1D7:  
:32218 :DIAG.WAIT:  
:32219 :  
:32220 :BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT  
:32221 :NAD/DIAG.WAIT1 ;BEFORE RETURNING TO DIAG.IDLE  
:32222 -----  
:32223 :159:  
:32224 :=*0*  
:32225 :DIAG.WAIT1:  
:32226 :  
:32227 :NAD/DIAG.IDLE ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND  
:32228 : ;COMMAND FROM THE CPU  
:32229 -----  
:32230 :15B:  
:32231 :=*1*  
:32232 :DIAG.WAIT2:  
:32233 :  
:32234 :NAD/DIAG.WAIT ;XFER GRANT HAS NOT BEEN ISSUED....  
:32235 : ;WAIT SOME MORE  
:32236 -----  
:32237 :WAIT.IDC.T39.3:  
U 1A65, DB00,1A :32238 :SKIP.IF[NO.FAST.INTERRUPT] ;WAIT FOR XFER REQ FROM IDC  
U 1A66, 89A6,84 :32239 :JMP [IDC.DONE.T39.3] ;XFER REQ SET  
U 1A67, 09A6,54 :32240 :JMP [WAIT.IDC.T39.3] ;WAIT SOME MORE  
:32241 :  
:32242 :IDC.DONE.T39.3:  
U 1A68, B700,15 :32243 :MOV LS[SETCSR.F0] TO WR[0] ;CLEAR CSR F<2:0>  
U 1A69, 3712,95 :32244 :MOV LS[SETCSR.CRDY] TO WR[1] ;SET CRDY TO ALLOW INTERRUPTS  
U 1A6A, AEC2,15 :32245 :BIS WR[1] TO WR[0]  
U 1A6B, 17A0,15 :32246 :PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR  
U 1A6C, 90FA,15 :32247 :MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ  
U 1A6D, DB00,15 :32248 :NOP ;ALLOW XFER REQ TO CLEAR  
U 1A6E, DB00,15 :32249 :NOP  
U 1A6F, DB00,15 :32250 :NOP  
U 1A70, 89A7,57 :32251 :JMP.IF[NO.INTERRUPT] TO [IE.OK.T39] ;JMP IF INT REQ DID NOT ASSERT  
U 1A71, 2F80,15 :32252 :CLR WR[0] ;FORCE AN ERROR  
U 1A72, 3640,95 :32253 :MOV LS[#1] TO WR[1]  
U 1A73, 0869,3C :32254 :JSR [CHECK.RESULT] ;REPORT THE ERROR  
U 1A74, 89A5,B4 :32255 :JMP [LOOP.T39.4] ;LOOP HERE IF ERR & LOE IS SET  
:32256 :  
:32257 :IE.OK.T39:  
U 1A75, 2F80,15 :32258 :CLR WR[0] ;CALL CHECK RESULT WITH GOOD  
U 1A76, 2F82,95 :32259 :CLR WR[1] ;DATA TO LOCK IN INTERMITTANT  
U 1A77, 0869,3C :32260 :JSR [CHECK.RESULT] ;ERRORS  
U 1A78, 89A5,B4 :32261 :JMP [LOOP.T39.4] ;LOOP HERE IF LOOP IS SET  
:32262 :  
U 1A79, FF82,15 :32263 :INC LS[ERROR.NUMBER] ;ERROR 5  
:32264 :  
:32265 :SET INTERRUPT REQUEST  
:32266 :
```

ZZ-ENKCF-1.4 :
: ENKCF.MCR
: ENKCF.MIC

ENKCF.MIC

Test 39 INTERRUPT R15-MAR-83

Fiche 4 Frame G3

Sequence 650

Rev 01.40 Page 644

MICRO2 1M(01) 15-MAR-83 11:46:22
Test 39 INTERRUPT REQUEST TEST 1 (M8388 IDC MODULE)

ENKCF Micro-diagnostic for IDC module

```
U 1A7A, 370E,15 :32267      MOV LS[SETCSR.F7] TO WR[0]      ;SET UP THE CSR TO BRANCH TO
U 1A7B, B720,95 :32268      MOV LS[SETCSR.MAINT] TO WR[1]      ; THE CORRECT IDC ROUTINE
U 1A7C, AEC2,15 :32269      BIS WR[1] TO WR[0]
U 1A7D, 3712,95 :32270      MOV LS[SETCSR.CRDY] TO WR[1]      ;SET CRDY
U 1A7E, AEC2,15 :32271      BIS WR[1] TO WR[0]
U 1A7F, B710,95 :32272      MOV LS[SETCSR.IE] TO WR[1]      ;SET INTERRUPT ENABLE
U 1A80, AEC2,15 :32273      BIS WR[1] TO WR[0]
U 1A81, 17A0,15 :32274      PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:32275      ;MAINT=1,CRDY=1,F<2:0>=111
:32276      SET.INT
:32277      -----
:32278      -----
:32279      *****
:32280      * IDC *
:32281      *****
:32282      -----
:32283      018:
:32284      =000
:32285      DIAG.IDLE:
:32286
:32287      BEN0/F0,
:32288      BEN1/F1,      ;BRANCH ON THE CSR F<2:0> BITS
:32289      BEN2/F2,
:32290      NAD/DIAG.IDLE
:32291      -----
:32292      01F:
:32293      =111
:32294
:32295      BEN0/CRDY.L,
:32296      BEN2/MAINT,      ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:32297      NAD/TEST.FUNC7      ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:32298      -----
:32299      07C:
:32300      =1*0
:32301      SET.INT:
:32302
:32303      FUNC/SET.INT,      ;SET INT FOR INT TEST...CRDY MUST
:32304      ; BE SET IN CSR
:32305      NAD/XFER.REQ
:32306      -----
:32307      1D8:
:32308      XFER.REQ:
:32309      UCMD/SET.XFER.RQST,      ;SEND XFER REQ TO THE CPU
:32310      NAD/DIAG.WAIT
:32311      -----
:32312      1D7:
:32313      DIAG.WAIT:
:32314
:32315      BEN1/XFER.REQ,      ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:32316      NAD/DIAG.WAIT1      ; BEFORE RETURNING TO DIAG.IDLE
:32317      -----
:32318      159:
:32319      =*0*
:32320      DIAG.WAIT1:
:32321
```

ZZ-
:
:

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

U 1

[illegible]

```

U 1A9F, A100,35 :32377      DT(LONG)&SET.ALU.CC
U 1AA0, 09A9,F1 :32378      JMP.IF[NEQ] TO [WAIT.BR.T39.1]      ;WAIT SOME SOME
U 1AA1, 09A3,F4 :32379      JMP [LOOP1.T39.3]                  ;ERROR LOOP
:32380
:32381
:32382      CLEANUP.T39:
:32383
U 1AA2, 97C4,15 :32384      PORT.CONTROL [RESET.BR]          ;CLEAR ALL INTERRUPT REQUESTS
:32385
U 1AA3, B788,15 :32386      MOV LS[#3] TO WR[0]                ;WAIT FOR THE INTERRUPT TO CLR
:32387
:32388      WAIT.BR.T39.3:
:32389      DEC WR[0],
U 1AA4, A100,35 :32390      DT(LONG)&SET.ALU.CC          ;WAIT SOME MORE
U 1AA5, 89AA,41 :32391      JMP.IF[NEQ] TO [WAIT.BR.T39.3]      ;WAIT SOME SOME
:32392
U 1AA6, DB00,15 :32393      NOP
U 1AA7, DB00,15 :32394      NOP
U 1AA8, DB00,15 :32395      NOP
U 1AA9, 8875,3C :32396      JSR [CLEANUP]          ;CLEANUP CONDITIONS SET BY
:32397      ; THIS TEST
:32398      END.T39:          ;END TEST
:32399
:32400
:32401
:32402

```

```

ZZ-1
:
:
U 1
U 1
U 1
U 1
U 1
U 1
U 1
U 1

```

```

:32403 .PAGE "Test 3A INTERRUPT REQUEST TEST 2 (M8388 IDC MODULE)"
:32404 :++
:32405 :
:32406 : Test Description:
:32407 :
:32408 :     This test is designed to insure that any of the Attention bits setting
:32409 :     will cause an interrupt if interrupts are enabled and CRDY is set.
:32410 :
:32411 : Assumptions:
:32412 :
:32413 :     It is assumed that all previous tests have run successfully.
:32414 :
:32415 : Test Steps:
:32416 :
:32417 :     1. Set up error mask, error number and module number in LS
:32418 :         (for error printouts) and clear the previous error number in LS.
:32419 :     2. Force IDC microcode to DIAG.IDLE.
:32420 :     3. Set bit <INTERUPT.EN> of the LS[CONTROL.STATUS] and allow time
:32421 :         any pending interrupts to clear.
:32422 :     4. Lower the IPL to 0. ( CLR LS[PSL.HW] )
:32423 :     5. Set IDC CSR <INT.IE>, interrupt enable and set CSR CRDY.
:32424 :     6. Select Drive 0.
:32425 :     7. Clear all of the attention bits.
:32426 :     8. Set up the CSR to branch to the correct IDC microcode routine.
:32427 :
:32428 :
:32429 :         *****
:32430 :         * IDC *
:32431 :         *****
:32432 :
:32433 :         IDC1. Execute an instruction with FUNC/SET.ATTN.
:32434 :         IDC2. Issue XFER.REQ to flag the CPU.
:32435 :
:32436 :     8. Wait for XFER.REQ from the IDC.
:32437 :     9. Clear CSR F<2:0> and send XFER GRANT to clear the XFER.REQ.
:32438 :    10. If Interrupt Request is not set then issue ERROR.
:32439 :    11. Issue port instruction to clear the BR 5.
:32440 :    12. Select the next drive number and go to Step 7 until all ATTN bits
:32441 :        have been tested.
:32442 :    13. Increment the error number.
:32443 :    14. Set all the Attention bits to set interrupt request.
:32444 :    15. INIT the IDC.
:32445 :    16. Check that the interrupt request cleared.
:32446 :    17. Clear IDC.
:32447 :    18. End test.
:32448 :
:32449 : Errors:
:32450 :
:32451 :     Printout:
:32452 :
:32453 :         EXPECTED      RECEIVED      OTHER DATA
:32454 :         N/A           N/A           ATTENTION BIT NUMBER
:32455 :
:32456 :     ERROR 1 - Setting the Attention bit did not set interrupt request.
:32457 :

```

:32458
:32459
:32460
:32461
:32462
:32463
:32464
:32465
:32466
:32467
:32468
:32469
:32470
:32471
:32472
:32473
:32474
:32475
:32476
:32477
:32478
:32479
:32480
:32481
:32482
:32483
:32484
:32485
:32486
:32487
:32488
:32489
:32490
:32491
:32492
:32493
:32494
:32495
:32496
:32497
:32498
:32499
:32500
:32501
:32502
:32503
:32504
:32505
:32506
:32507
:32508
:32509
:32510
:32511
:32512

: ERROR 2 - IDC INIT did not clear the interrupt request.
: Debug:
: ERROR 1-
: Any pending interrupts are cleared and the IPL is lowered to 0.
: One Attention Bit is set with CSR <CRDY>=1 and interrupts are enabled by
: setting CSR <IE> bit 6. This will assert SET INT REQ H at the ATTN REG
: PAL. This is an input to the FIFOA ADRES CNTR LO PAL. The inputs to this
: PAL should now be:
: SET INT REQ H = H
: RESET BR L = H
: IE H = H
: CRDY H = H
: This combination of inputs will assert INT REQ DIN H. This is an input
: to the INIT FF and at the next P2 CLOCK L, INT REQ H asserts. INT REQ H
: is inverted and leveled the board as UBUS BR5 L.
: This process is repeated with setting each of the ATTENTION BITS.
: ERROR 2-
: The CPU issues PORT.CONTROL [CLEAR.IDC] instructions which are decoded
: by the PORT RD/WR DECODE PAL. This will assert CLEAR IDC L. At the
: next P2 CLOCK L, the INIT FF will assert INIT SYNC 1 H. At the next
: clock INIT H is asserted. This is an input to the ATTN REG PAL and when
: INIT H asserts, SET INT REQ H will clear.
: --
: *****
: T.3A:
: MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT 15 IN WR[0] FOR
: ; CONTROL AND STATUS WORD
: MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND
: ; STATUS WORD - BIT 15
: ; INDICATES START OF TEST TO
: ; CONSOLE
: ; ISSUE CPU ATTN TO CONSOLE
: MISC [SET.CP.ATTN]
: WAIT.T3A.0:
: JMP [WAIT.T3A.0] ;LOOP HERE WAITING FOR
: ; CONSOLE TO RESPOND
: CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
: ; IN LS
: MOV LS[IDC] TO WR[0] ;STORE MODULE CODE IN LS TO
: MOV WR[0] TO LS[MODULE.NUM] ; INDICATE IDC MODULE
: MOV LS[#1] TO WR[0] ;SET WRO = 1
: MOV WR[0] TO LS[ERROR.NUMBER] ;SET UP ERROR NUMBER = 1 IN LS
: CLR LS[ERROR.MASK] ;TO FORCE ERRORS
: JSR [DIAG.CODE] ;INIT THE IDC AND FORCE IT
: ; TO DIAG.IDLE

U 1AAA, B65E,15
U 1AAB, 3E80,15

U 1AAC, 10E0,15
U 1AAD, 89AA,D4

U 1AAE, 65A0,15

U 1AAF, B64A,15
U 1AB0, 3E8C,15

U 1AB1, B640,15
U 1AB2, BE82,15

U 1AB3, E58A,15
U 1AB4, 8872,EC

ZZ-1
:
:

U 1
U 1

U 1
U 1

U 1
U 1

```

U 1AB5, 3660,15 :32513      MOV LS[INTERUPT.EN] TO WR[0]      ;CLEAR ANY PENDING INTERRUPTS
U 1AB6, 3E80,15 :32514      MOV WR[0] TO LS[CONTROL.STATUS]
:32515
U 1AB7, 10E0,15 :32516      MISC [SET.CP.ATTN]      ;ISSUE CPU ATTN TO CONSOLE
:32517      WAIT.T3A.1:
U 1AB8, 09AB,84 :32518      JMP [WAIT.T3A.1]      ;LOOP HERE WAITING FOR
:32519      ; CONSOLE TO RESPOND
:32520
U 1AB9, B670,15 :32521      MOV LS[OTHER.DATA] TO WR[0]      ;SET UP TO PRINT OTHER DATA
U 1ABA, 3E80,15 :32522      MOV WR[0] TO LS[CONTROL.STATUS]
:32523
U 1ABB, B66E,15 :32524      MOV LS[NA.EXP.REC] TO WR[0]      ;EXP AND REC DATA ARE NOT
U 1ABC, 6D80,15 :32525      BIS WR[0] TO LS[CONTROL.STATUS]      ; APPLICABLE
:32526
:32527
U 1ABD, 2F80,15 :32528      CLR WR[0]
U 1ABE, BFFE,15 :32529      MOV WR[0] TO LS[PSL.HW]      ;LOWER THE IPL TO 0
:32530
:32531      ;WR[3] WILL SAVE THE DRIVE NUMBER FOR PRINTOUTS
:32532      ;WR[2] WILL HAVE THE DRIVE SELECT IN BITS<9:8>
:32533
U 1ABF, 2F87,95 :32534      CLR WR[3]      ;SAV DRV NUM FOR PRINTOUTS
U 1AC0, 3715,15 :32535      MOV LS[SETCSR.DS0] TO WR[2]      ;SELECT DRIVE 0
:32536
:32537      ;SET UP THE CSR TO SET ATTN WITH INTERRUPTS ENABLED
:32538      LOOP.T3A.1:
:32539      TST.NXT.T3A:
:32540
U 1AC1, 370E,15 :32541      MOV LS[SETCSR.F7] TO WR[0]
U 1AC2, B7D6,95 :32542      MOV LS[#000FG000] TO WR[1]      ;CLEAR ALL ATTN BITS
U 1AC3, AEC2,15 :32543      BIS WR[1] TO WR[0]
U 1AC4, 17A0,15 :32544      PORT.WRITE PORT.ADRS[CSR] WITH WR[0]      ;WRITE THE CSR
:32545      ;MAINT=0,CRDY=0,F<2:0>=111
:32546      ;CONTINUE TO BRANCH
U 1AC5, 379E,15 :32547      MOV LS[SETCSR.R80] TO WR[0]      ;SET UP THE CSR TO CONTINUE TO
U 1AC6, B708,95 :32548      MOV LS[SETCSR.F4] TO WR[1]      ; BRANCH TO THE CORRECT ROUTINE
U 1AC7, AEC2,15 :32549      BIS WR[1] TO WR[0]
U 1AC8, AEC4,15 :32550      BIS WR[2] TO WR[0]      ;INCLUDE THE DRIVE SELECT
U 1AC9, 3712,95 :32551      MOV LS[SETCSR.CRDY] TO WR[1]      ;SET CRDY TO ALLOW INTERRUPTS
U 1ACA, AEC2,15 :32552      BIS WR[1] TO WR[0]
U 1ACB, B710,95 :32553      MOV LS[SETCSR.IE] TO WR[1]      ;ENABLE INTERRUPTS
U 1ACC, AEC2,15 :32554      BIS WR[1] TO WR[0]
U 1ACD, 17A0,15 :32555      PORT.WRITE PORT.ADRS[CSR] WITH WR[0]      ;WRITE THE CSR
:32556      ;F<2:0>=100
:32557      ;USET ATTN
:32558      -----
:32559      -----
:32560      ;
:32561      ; *****
:32562      ; * IDC *
:32563      ; *****
:32564      ;
:32565      ;018:
:32566      ;=000
:32567      ;DIAG.IDLE:
:

```



```

32568 : BEN0/F0,
32569 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
32570 : BEN2/F2,
32571 : NAD/DIAG.IDLE
32572 :-----
32573 : 01F:
32574 : =111
32575 :
32576 : BEN0/CRDY.L,
32577 : BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
32578 : NAD/TEST.FUNC7 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
32579 :-----
32580 : 079:
32581 : =0*1
32582 :
32583 : NAD/GROUP1
32584 :-----
32585 : 182:
32586 : =0**
32587 : GROUP1:
32588 :
32589 : BEN2/R80.FORMAT.H, ;WHEN THE R80 FORMAT BIT SETS..BRANCH
32590 : NAD/GROUP1 ; TO NEXT WORD AND THEN BRANCH ON THE
32591 : ; FUNCTION BITS
32592 :-----
32593 : 186:
32594 : =1**
32595 :
32596 : BEN0/F0,
32597 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
32598 : BEN2/F2,
32599 : NAD/GROUP1.F0
32600 :-----
32601 : 02C:
32602 : =100
32603 : USET.ATTN:
32604 :
32605 : FUNC/SET.ATTN, ;SET ATTENTION BE SURE CRDY SET TO ALLOW
32606 : ; INTERRUPTS
32607 : NAD/XFER.REQ ;SEND XFER REQ TO FLAG THE CPU
32608 :-----
32609 : 1D8:
32610 : XFER.REQ:
32611 : UCMD/SET.XFER.RQST, ;SEND XFER REQ TO THE CPU
32612 : NAD/DIAG.WAIT
32613 :-----
32614 : 1D7:
32615 : DIAG.WAIT:
32616 :
32617 : BEN1/XFER.REQ, ;WAIT FOR THE CPU TO ISSUE XFER GRANT
32618 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
32619 :-----
32620 : 159:
32621 : =*0*
32622 : DIAG.WAIT1:

```

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100

U 1B
U 1B

U 1E
U 1E

U 1B
U 1E
U 1E

U 1E
U 1E

U 1E
U 1E
U 1E

U 1E
U 1E
U 1E
U 1E
U 1E

```

32678
U 1AEB, 4051,15 :32679      ADD LS[#100] TO WR[2]      ;SELECT THE NEXT DRIVE
U 1AEC, 89AC,14 :32680      JMP [TST.NXT.T3A]      ;GO TEST THE NEXT BIT
:32681
:32682 TST.INIT.T3A:
:32683
U 1AED, 3642,15 :32684      MOV LS[#2] TO WR[0]
U 1AEE, BE82,15 :32685      MOV WR[0] TO LS[ERROR.NUMBER]      ;ERROR 2
:32686
U 1AEF, B715,95 :32687      MOV LS[SETCSR.DS0] TO WR[3]
:32688
:32689 SET.ATTN.T3A:
:32690 ;SET ALL THE ATTENTION BITS
:32691
U 1AF0, 370E,15 :32692      MOV LS[SETCSR.F7] TO WR[0]
U 1AF1, 17A0,15 :32693      PORT.WRITE PORT.ADRS[CSR] WITH WR[0]      ;WRITE THE CSR
:32694      ;MAINT=0,CRDY=0,F<2:0>=111
:32695 ;CONTINUE TO BRANCH
U 1AF2, 379E,15 :32696      MOV LS[SETCSR.R80] TO WR[0]      ;SET UP THE CSR TO CONTINUE TO
U 1AF3, B708,95 :32697      MOV LS[SETCSR.F4] TO WR[1]      ; BRANCH TO THE CORRECT ROUTINE
U 1AF4, AEC2,15 :32698      BIS WR[1] TO WR[0]
U 1AF5, 3712,95 :32699      MOV LS[SETCSR.CRDY] TO WR[1]      ;SET CRDY
U 1AF6, AEC2,15 :32700      BIS WR[1] TO WR[0]
U 1AF7, B710,95 :32701      MOV LS[SETCSR.IE] TO WR[1]      ;ENABLE INTERRUPTS
U 1AF8, AEC2,15 :32702      BIS WR[1] TO WR[0]
U 1AF9, 2EC6,15 :32703      BIS WR[3] TO WR[0]      ;INCLUDE THE DRIVE SELECT
U 1AFA, 17A0,15 :32704      PORT.WRITE PORT.ADRS[CSR] WITH WR[0]      ;WRITE THE CSR
:32705      ;F<2:0>=100
:32706 ;USET ATTN
:32707 -----
:32708 -----
:32709      *****
:32710      * IDC *
:32711      *****
:32712 -----
:32713 ;018:
:32714 ;=000
:32715 ;DIAG.IDLE:
:32716
:32717      BEN0/F0,
:32718      BEN1/F1,      ;BRANCH ON THE CSR F<2:0> BITS
:32719      BEN2/F2,
:32720      NAD/DIAG.IDLE
:32721 -----
:32722 ;01F:
:32723 ;=111
:32724
:32725      BEN0/CRDY.L,
:32726      BEN2/MAINT,      ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:32727      NAD/TEST.FUNC7      ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:32728 -----
:32729 ;079:
:32730 ;=0*1
:32731
:32732      NAD/GROUP1
  
```

```

32733 -----
32734 :182:
32735 : =0**
32736 :GROUP1:
32737
32738 :      BEN2/R80.FORMAT.H,          ;WHEN THE R80 FORMAT BIT SETS..BRANCH
32739 :      NAD/GROUP1                  ; TO NEXT WORD AND THEN BRANCH ON THE
32740 :                                  ; FUNCTION BITS
32741 -----
32742 :186:
32743 : =1**
32744
32745 :      BEN0/F0,
32746 :      BEN1/F1,                    ;BRANCH ON THE CSR F<2:0> BITS
32747 :      BEN2/F2,
32748 :      NAD/GROUP1.F0
32749 -----
32750 :02C:
32751 : =100
32752 :USET.ATTN:
32753
32754 :      FUNC/SET.ATTN,              ;SET ATTENTION BE SURE CRDY SET TO ALLOW
32755 :                                  ; INTERRUPTS
32756 :      NAD/XFER.REQ               ;SEND XFER REQ TO FLAG THE CPU
32757 -----
32758 :1D8:
32759 :XFER.REQ:
32760 :      UCMD/SET.XFER.RQST,         ;SEND XFER REQ TO THE CPU
32761 :      NAD/DIAG.WAIT
32762 -----
32763 :1D7:
32764 :DIAG.WAIT:
32765
32766 :      BEN1/XFER.REQ,              ;WAIT FOR THE CPU TO ISSUE XFER GRANT
32767 :      NAD/DIAG.WAIT1             ; BEFORE RETURNING TO DIAG.IDLE
32768 -----
32769 :159:
32770 : =*0*
32771 :DIAG.WAIT1:
32772
32773 :      NAD/DIAG.IDLE              ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
32774 :                                  ; COMMAND FROM THE CPU
32775 -----
32776 :15B:
32777 : =*1*
32778 :DIAG.WAIT2:
32779
32780 :      NAD/DIAG.WAIT              ;XFER GRANT HAS NOT BEEN ISSUED....
32781 :                                  ; WAIT SOME MORE
32782 -----
32783 :WAIT.IDC.T3A.2:
32784 :      SKIP.IF[NO.FAST.INTERRUPT]  ;WAIT FOR XFER REQ FROM IDC
32785 :      JMP [IDC.DONE.T3A.2]        ;XFER REQ SET
32786 :      JMP [WAIT.IDC.T3A.2]        ;WAIT SOME MORE
32787

```

U 1AFB, DB00, 1A
U 1AFC, 89AF, E4
U 1AFD, 89AF, B4

U U U U U U U U U U

ZZ-ENKCF-1.4 :
: ENKCF.MCR
: ENKCF.MIC

ENKCF.MIC

MICRO2 1M(01)

Test 3A INTERRUPT 15-MAR-83

15-MAR-83 11:46:22

Test 3A INTERRUPT REQUEST TEST 2 (M8388 IDC MODULE)

Fiche 4 Frame D4

Sequence 660

Page 654

ZZ-
:
:

```
U 1AFE, B700,15 :32788 IDC.DONE.T3A.2:
U 1AFF, 3712,95 :32789 MOV LS[SETCSR.F0] TO WR[0] ;CLEAR CSR F<2:0>
U 1B00, AEC2,15 :32790 MOV LS[SETCSR.CRDY] TO WR[1] ;SET CRDY
U 1B01, B710,95 :32791 BIS WR[1] TO WR[0]
U 1B02, AEC2,15 :32792 MOV LS[SETCSR.IE] TO WR[1] ;ENABLE INTERRUPTS
U 1B03, 17A0,15 :32793 BIS WR[1] TO WR[0]
U 1B04, 90FA,15 :32794 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:32795 MISC2 [TRANSFER.GRANT] ;SEND GRANT TO CLR THE XFER REQ
:32796
U 1B05, C051,95 :32797 ADD LS[#100] TO WR[3] ;SELECT THE NEXT DRIVE
:32798 CMP LS[#400] WITH WR[3], ;ALL BITS BEEN SET?
U 1B06, 4E55,B5 :32799 DT(LONG)&SET.ALU.CC
U 1B07, 09AF,01 :32800 JMP.IF[NEQ] TO [SET.ATTN.T3A] ;JMP IF NO
:32801
:32802 LOOP.T3A.2:
U 1B08, 17CC,15 :32803 PORT.CONTROL [CLEAR.IDC] ;INIT THE IDC
U 1B09, 17CC,15 :32804 PORT.CONTROL [CLEAR.IDC]
U 1B0A, 17CC,15 :32805 PORT.CONTROL [CLEAR.IDC]
:32806
U 1B0B, B788,15 :32807 MOV LS[#3] TO WR[0] ;WAIT FOR INT TO CLR
:32808 WAIT.INIT.T3A:
:32809 DEC WR[0], ;WAIT SOME MORE
U 1B0C, A100,35 :32810 DT(LONG)&SET.ALU.CC
U 1B0D, 89B0,C1 :32811 JMP.IF[NEQ] TO [WAIT.INIT.T3A] ;WAIT SOME MORE
:32812
:32813
U 1B0E, 89B1,47 :32814 JMP.IF[NO.INTERRUPT] TO [GOOD.LOOP.T3A] ;JMP IF OK
U 1B0F, 2F80,15 :32815 CLR WR[0] ;FORCE AN ERROR
U 1B10, 3640,95 :32816 MOV LS[#1] TO WR[1]
U 1B11, 6588,15 :32817 CLR LS[ADDRESS.DATA] ;CLEAR OTHER DATA
U 1B12, 0869,3C :32818 JSR [CHECK.RESULT] ;REPORT THE ERROR
U 1B13, 09B0,84 :32819 JMP [LOOP.T3A.2] ;LOOP HERE ON ERR & LOE IS SET
:32820 GOOD.LOOP.T3A:
:32821
U 1B14, 2F80,15 :32822 CLR WR[0] ;CALL CHECK RESULT WITH GOOD
U 1B15, 2F82,95 :32823 CLR WR[1] ; DATA TO LOCK IN INTERMITTANT
U 1B16, 6588,15 :32824 CLR LS[ADDRESS.DATA] ; ERRORS
U 1B17, 0869,3C :32825 JSR [CHECK.RESULT]
U 1B18, 09B0,84 :32826 JMP [LOOP.T3A.2] ;LOOP HERE IF LOOP IS SET
:32827
:32828 CLEANUP.T3A:
U 1B19, 97C4,15 :32829 PORT.CONTROL [RESET.BR] ;CLEAR ALL INTERRUPTS
U 1B1A, B788,15 :32830 MOV LS[#3] TO WR[0] ;WAIT FOR INT TO CLR
:32831 WAIT.BR.T3A.1:
:32832 DEC WR[0], ;WAIT SOME MORE
U 1B1B, A100,35 :32833 DT(LONG)&SET.ALU.CC
U 1B1C, 89B1,B1 :32834 JMP.IF[NEQ] TO [WAIT.BR.T3A.1] ;WAIT SOME MORE
:32835
U 1B1D, 8875,3C :32836 JSR [CLEANUP] ;CLEANUP THE CONDITIONS SET
:32837 ; BY THIS TEST
:32838 END.T3A: ;END TEST
:32839
:32840
:32841
:32842
```

U 1

U 1

U 1

U 1

U 1

U 1

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

U

32843 : PAGE "Test 3B AUTOMODE TEST (M8388 IDC MODULE)"

32844 : ++

32845 :

32846 : Test Description:

32847 :

32848 : This test is designed to insure that when in automode, after one READ
 32849 : DATA instruction, the IDC continues to send the CPU new read data when
 32850 : a MOV PORT TO LS[] instruction is issued. It will also verify that
 32851 : when automode is cleared, one more MOV PORT TO LS[] reads new data and
 32852 : then the IDC does not continue to send new read data.

32853 : Automode can only do longword reads.

32854 : The data in FIFO A will be:

32855 : Address 0 - 3: 11111111

32856 : Address 4 - 7: 22222222

32857 : Address 8 - B: 33333333

32858 : Address C - F: 44444444

32859 : Address 10 - 13: 55555555

32860 :

32861 : Automode will be enabled and then the first longword read (11111111)
 32862 : will be done by first issuing a READ.DATA WORD and then doing a
 32863 : MOV.PORT to LS[]. Then a MOV.PORT to LS[] instruction will be issued.
 32864 : This should move the next longword of data (22222222) from FIFO A to
 32865 : LS[]. Another MOV.PORT TO LS[] will be issued to insure that automode
 32866 : enabled all necessary conditions. This should move the next longword of
 32867 : data (33333333) from FIFO A to LS[].

32868 : Automode will then be disabled. A MOV.PORT TO LS[] instruction will be
 32869 : issued. This should read the next longword of data (44444444). Then
 32870 : one more MOV.PORT to LS[] will be issued and the next longword
 32871 : (55555555) should not be read.

32872 : The CPU will then read the data in LS to insure that automode worked
 32873 : correctly.

32874 :

32875 : Assumptions:

32876 :

32877 : It is assumed that all previous tests have run successfully.

32878 :

32879 : Test Steps:

32880 :

- 32881 : 1. Set up error mask, error number and module number in LS
- 32882 : (for error printouts) and clear the previous error number in LS.
- 32883 : 2. INIT the IDC and force it to DIAG.IDLE.
- 32884 : 3. Write the data to the FIFO.
- 32885 : 4. Set automode.
- 32886 : 5. Read the first location of the FIFO and MOV.PORT TO LS[].
- 32887 : 6. Issue 2 MOV PORT TO LS[] instructions.
- 32888 : 7. Clear automode.
- 32889 : 8. Issue 2 MOV PORT TO LS[] instructions.
- 32890 : 9. Check the five locations in LS.
- 32891 : 10. Clear IDC.
- 32892 : 11. End test.

32893 :

32894 :

32895 : Errors:

32896 :

32897 : Printout:

U 1
U 1
U 1
U 1

U 1
U 1

U 1
U 1
U 1
U 1

U 1

U 1

U

U

U
U

U
U

U
U

```

:32953 : new longword of data is read.
:32954 :
:32955 : ERROR 5:
:32956 : AUTOMODE is cleared when the CPU issues a CLEAR.AUTO instruction.
:32957 : This instruction will be decoded in the PORT INTERFACE REG PAL.
:32958 : The inputs should be:
:32959 :     CSR 17 H = H
:32960 :     CSR 14 H = H
:32961 :     CSR 12 H = H
:32962 :     CSR 11 H = L
:32963 :     CSR 10 H = H
:32964 :     PORT INSTR H = H
:32965 :     CPU P2 H = H
:32966 : This will deassert AUTOMODE H, and new data should not be returned
:32967 : when only a MOV PORT TO LS[] is issued.
:32968 : --
:32969 : *****
:32970 : T.3B:
U 1B1E, B65E,15 :32971 : MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT 15 IN WR[0] FOR
:32972 : ; CONTROL AND STATUS WORD
U 1B1F, 3E80,15 :32973 : MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND
:32974 : ; STATUS WORD - BIT 15
:32975 : ; INDICATES START OF TEST TO
:32976 : ; CONSOLE
U 1B20, 10E0,15 :32977 : MISC [SET.CP.ATTN] ;ISSUE CPU ATTN TO CONSOLE
:32978 : WAIT.T3B.0:
U 1B21, 89B2,14 :32979 : JMP [WAIT.T3B.0] ;LOOP HERE WAITING FOR
:32980 : ; CONSOLE TO RESPOND
:32981 :
U 1B22, 65A0,15 :32982 : CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
:32983 : ; IN LS
:32984 :
U 1B23, B64A,15 :32985 : MOV LS[IDC] TO WR[0] ;STORE MODULE CODE IN LS TO
U 1B24, 3E8C,15 :32986 : MOV WR[0] TO LS[MODULE.NUM] ; INDICATE IDC MODULE
:32987 :
:32988 :
:32989 :
U 1B25, E58A,15 :32990 : CLR LS[ERROR.MASK] ;SET UP ERROR MASK TO CHECK
:32991 : ; ALL BITS
U 1B26, 8872,EC :32992 : JSR [DIAG.CODE] ;INIT THE IDC AND FORCE IT TO
:32993 : ; DIAG.IDLE
U 1B27, 17D8,15 :32994 : PORT.CONTROL [SEL.FIFO.A] ;SELECT FIFO A
U 1B28, 17C0,15 :32995 : PORT.CONTROL [CLEAR.FIFO.CNTR] ;SET ADDRESS = 0
:32996 :
U 1B29, 37D0,15 :32997 : MOV LS[#11111111] TO WR[0] ;WRITE THE FIRST DATA
U 1B2A, 17AC,15 :32998 : PORT.WRITE PORT.ADRS[DATA.WORD] WITH WR[0] ;WRITE A LONGWORD OF DATA
:32999 :
U 1B2B, B7D2,15 :33000 : MOV LS[#22222222] TO WR[0] ;WRITE NEXT LONGWORD OF DATA
U 1B2C, 17AC,15 :33001 : PORT.WRITE PORT.ADRS[DATA.WORD] WITH WR[0] ;WRITE A LONGWORD OF DATA
:33002 :
U 1B2D, B6C4,15 :33003 : MOV LS[#33333333] TO WR[0] ;WRITE NEXT LONGWORD OF DATA
U 1B2E, 17AC,15 :33004 : PORT.WRITE PORT.ADRS[DATA.WORD] WITH WR[0] ;WRITE A LONGWORD OF DATA
:33005 :
U 1B2F, B7D4,15 :33006 : MOV LS[#44444444] TO WR[0] ;WRITE NEXT LONGWORD OF DATA
U 1B30, 17AC,15 :33007 : PORT.WRITE PORT.ADRS[DATA.WORD] WITH WR[0] ;WRITE A LONGWORD OF DATA
  
```



```

:33008
U 1B31, 369A,15 :33009      MOV LS[#55555555] TO WR[0]      ;WRITE NEXT LONGWORD OF DATA
U 1B32, 17AC,15 :33010      PORT.WRITE PORT.ADRS[DATA.WORD] WITH WR[0] ;WRITE A LONGWORD OF DATA
:33011
:33012      LOOP.T3B.1:
:33013
U 1B33, B640,15 :33014      MOV LS[#1] TO WR[0]      ;SET WRO = 1
U 1B34, BE82,15 :33015      MOV WR[0] TO LS[ERROR.NUMBER] ;SET UP ERROR NUMBER = 1 IN LS
:33016
U 1B35, 17C0,15 :33017      PORT.CONTROL [CLEAR.FIFO.CNTR] ;SET ADDRESS = 0
:33018
U 1B36, 97D0,15 :33019      PORT.CONTROL [SET.AUTO] ;SET AUTOMODE TO CONTINUE READS
:33020
U 1B37, 9090,15 :33021      MISC [SET.PORT.1.0] ;SELECT THE IDC
U 1B38, 978C,15 :33022      PORT.READ PORT.ADRS[DATA.WORD] ;SET UP TO READ A LONGWORD
U 1B39, DB00,15 :33023      NOP ;ALLOW TIME FOR IDC TO ASSEMBLE
:33024      ; A LONGWORD OF DATA
U 1B3A, 3B34,15 :33025      MOV PORT TO LS[PORT1] ;READ THE DATA FROM FIFO AND
U 1B3B, DB00,15 :33026      NOP ;SAVE IT IN LS
U 1B3C, BB36,15 :33027      MOV PORT TO LS[PORT2] ;READ THE NEXT LONGWORD
U 1B3D, DB00,15 :33028      NOP
U 1B3E, 3B38,15 :33029      MOV PORT TO LS[PORT3] ;READ THE NEXT LONGWORD
U 1B3F, DB00,15 :33030      NOP
U 1B40, 17D4,15 :33031      PORT.CONTROL [CLEAR.AUTO] ;CLEAR AUTOMODE
:33032
U 1B41, BB3A,15 :33033      MOV PORT TO LS[PORT4] ;SHOULD READ ONE MORE WORD
U 1B42, DB00,15 :33034      NOP
U 1B43, BB3C,15 :33035      MOV PORT TO LS[PORT5] ;SHOULD NOT SEND NEW DATA
:33036
:33037      ;CHECK THE DATA THAT WAS READ
:33038
U 1B44, 3734,15 :33039      MOV LS[PORT1] TO WR[0] ;SET UP THE RECEIVED DATA
U 1B45, B7D0,95 :33040      MOV LS[#11111111] TO WR[1] ;SET UP THE EXPECTED DATA
U 1B46, 0869,3C :33041      JSR [CHECK.RESULT] ;CHECK THE DATA
U 1B47, 89B3,34 :33042      JMP [LOOP.T3B.1] ;LOOP HERE IF ERR & LOE IS SET
:33043
:33044
U 1B48, FF82,15 :33045      INC LS[ERROR.NUMBER] ;ERROR 2
:33046
U 1B49, B736,15 :33047      MOV LS[PORT2] TO WR[0] ;SET UP THE RECEIVED DATA
U 1B4A, 37D2,95 :33048      MOV LS[#22222222] TO WR[1] ;SET UP THE EXPECTED DATA
U 1B4B, 0869,3C :33049      JSR [CHECK.RESULT] ;CHECK THE DATA
U 1B4C, 89B3,34 :33050      JMP [LOOP.T3B.1] ;LOOP HERE IF ERR & LOE IS SET
:33051
:33052
U 1B4D, FF82,15 :33053      INC LS[ERROR.NUMBER] ;ERROR 3
:33054
U 1B4E, 3738,15 :33055      MOV LS[PORT3] TO WR[0] ;SET UP THE RECEIVED DATA
U 1B4F, 36C4,95 :33056      MOV LS[#33333333] TO WR[1] ;SET UP THE EXPECTED DATA
U 1B50, 0869,3C :33057      JSR [CHECK.RESULT] ;CHECK THE DATA
U 1B51, 89B3,34 :33058      JMP [LOOP.T3B.1] ;LOOP HERE IF ERR & LOE IS SET
:33059
:33060
U 1B52, FF82,15 :33061      INC LS[ERROR.NUMBER] ;ERROR 4
:33062
  
```

```

U 1B53, B73A,15 ;33063      MOV LS[PORT4] TO WR[0]      ;SET UP THE RECEIVED DATA
U 1B54, 37D4,95 ;33064      MOV LS[#44444444] TO WR[1]    ;SET UP THE EXPECTED DATA
U 1B55, 0869,3C ;33065      JSR [CHECK.RESULT]        ;CHECK THE DATA
U 1B56, 89B3,34 ;33066      JMP [LOOP.T3B.1]           ;LOOP HERE IF ERR & LOE IS SET
                  ;33067
                  ;33068
U 1B57, FF82,15 ;33069      INC LS[ERROR.NUMBER]        ;ERROR 5
                  ;33070
U 1B58, B73C,15 ;33071      MOV LS[PORT5] TO WR[0]      ;SET UP THE RECEIVED DATA
U 1B59, 37D4,95 ;33072      MOV LS[#44444444] TO WR[1]    ;SET UP THE EXPECTED DATA
U 1B5A, 0869,3C ;33073      JSR [CHECK.RESULT]        ;CHECK THE DATA
U 1B5B, 89B3,34 ;33074      JMP [LOOP.T3B.1]           ;LOOP HERE IF ERR & LOE IS SET
                  ;33075
U 1B5C, 8875,3C ;33076      JSR [CLEANUP]                ;CLEANUP THE CONDITIONS SET
                  ;33077      ;BY THIS TEST
                  ;33078      ;END TEST
                  ;33079
                  ;33080
                  ;33081
  
```

END.T3B:

```

:33082 .PAGE "Test 3C TIMEOUT TEST 1 (M8388 IDC MODULE)"
:33083 .++
:33084
:33085 .Test Description:
:33086
:33087 .This test will check that the timeout counter times out within the
:33088 .specified amount of time which is 150 ms +50ms and -50ms after CSR
:33089 .<CRDY> is cleared.
:33090 .This test will also insure that a UCDM/CLEAR.TIMEOUT clears
:33091 .timeouts.
:33092 .It will also check that a timeout causes the IDC to JAM and return
:33093 .to the IDC Microcode Idle Loop.
:33094
:33095 .This test cannot be single-stepped because the timeout is generated
:33096 .from a one-shot.
:33097
:33098 .Assumptions:
:33099
:33100 .It is assumed that all previous tests have been run successfully.
:33101
:33102 .Test Steps:
:33103
:33104 .1. Set up error mask, error number and module number in LS
:33105 .(for error printouts) and clear the previous error number in LS.
:33106 .2. Force the IDC to DIAG.IDLE.
:33107 .3. Clear CSR <CRDY> to start timer.
:33108 .4. Set up a loop to wait 200 ms.
:33109 .5. Check CSR <OPI> (bit 10) = 1 to insure that IDC timed out.
:33110 .6. Increment the error number.
:33111 .7. To insure that the IDC did JAM when it timed out,
:33112 .check CSR <CRDY> = 1. (The IDC microcode should JAM, clear the
:33113 .timeout, then set CSR <CRDY> and then return to the IDC IDLE LOOP).
:33114 .8. Increment the error number.
:33115 .9. Clear CSR <CRDY> with CSR <MAINT> set to clear OPI.
:33116 .10. Check CSR <OPI> (bit 10) remains clear to insure that
:33117 .timeout cleared.
:33118 .11. Increment the error number.
:33119 .12. Force the IDC to DIAG.IDLE.
:33120 .13. Clear CSR <CRDY>.
:33121 .14. Set up a loop to wait 100 ms.
:33122 .15. Check CSR <OPI> (bit 10) = 0 to insure that IDC did not time out.
:33123 .16. Clear IDC.
:33124 .17. End test.
:33125
:33126 .Errors:
:33127
:33128 .Printout:
:33129 .
:33130 .EXPECTED RECEIVED
:33131 .CSR CSR
:33132 .<OPI> <OPI>
:33133 .
:33134 .ERROR 1 - The IDC did not timeout within the specified time after
:33135 .INIT was issued and then CSR <CRDY> was cleared.
:33136 .
:33137 .ERROR 2 - A timeout did not JAM the IDC.

```

```

U 1
U 1
U 1
U 1

```

```

:33137 :
:33138 : ERROR 3 - UCMD/CLEAR.TIMEOUT did not clear the timeout.
:33139 :
:33140 : ERROR 4 - The IDC timeout out too soon after CSR <CRDY> was cleared.
:33141 :
:33142 : Debug:
:33143 :
:33144 : ERROR 1-
:33145 : When CSR <CRDY> is cleared with URESET TIMER L = H, the one-shot
:33146 : is set up and should trigger 150ms (+50ms or -50ms) later if the
:33147 : set-up conditions remain unchanged. The test then waits 200ms for the
:33148 : timeout to occur. When the one-shot is triggered, TIMER H will assert.
:33149 : This signal is an input to the CONTROL STATUS REG PAL. This should
:33150 : assert TIMEOUT H. TIMEOUT H = H in that same PAL will assert OPI L.
:33151 : When the CPU reads the CSR, OPI L = L is fed through an 8-line DRIVER
:33152 : and becomes BUS IO 10 H = H.
:33153 :
:33154 : ERROR 2-
:33155 : After TIMEOUT H asserts in the CONTROL STATUS REG PAL, (see ERROR 1),
:33156 : on the next P2 CLOCK L, TIMEOUT H is fed through the INIT FF and
:33157 : JAM L should assert and is then inverted and used as JAM H. JAM H = H
:33158 : will cause the microsequencer to jam to microstate FF. The micocode
:33159 : will then select BEN2/OPI L. OPI L will be asserted so the micocode
:33160 : will branch to a state that will set CSR <CRDY> and then go to the
:33161 : IDC IDLE LOOP.
:33162 :
:33163 : ERROR 3-
:33164 : After the TIMEOUT has occurred and the IDC has JAMMED, the CSR will
:33165 : be set up to branch to an instruction that will clear the timeout.
:33166 : In this micro-instruction UCMD<2:0> = 111. The UCMD bits will be
:33167 : decoded in the CONTROL STATUS REG PAL and should clear TIMEOUT H
:33168 : which in turn will clear OPI L.
:33169 :
:33170 : ERROR 4 -
:33171 : When CSR <CRDY> is cleared with URESET TIMER L = H, the one-shot
:33172 : is set up and should trigger 150ms (+50ms or -50ms) later if the
:33173 : set-up conditions remain unchanged. The test then waits 100ms for the
:33174 : timeout to occur. If the one-shot is triggered, TIMER H will assert.
:33175 : CSR <OPI> is then checked to insure that the one-shot did not trigger
:33176 : too soon. (See ERROR 1)
:33177 : --
:33178 : *****
:33179 : T.3C:
U 1B5D, B65E,15 :33180 : MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT 15 IN WR[0] FOR
:33181 : ; CONTROL AND STATUS WORD
U 1B5E, 3E80,15 :33182 : MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND
:33183 : ; STATUS WORD - BIT 15
:33184 : ; INDICATES START OF TEST TO
:33185 : ; CONSOLE
U 1B5F, 10E0,15 :33186 : MISC [SET.CP.ATTN] ;ISSUE CPU ATTN TO CONSOLE
:33187 : WAIT.T3C.0:
U 1B60, 89B6,04 :33188 : JMP [WAIT.T3C.0] ;LOOP HERE WAITING FOR
:33189 : ; CONSOLE TO RESPOND
:33190 :
U 1B61, 65A0,15 :33191 : CLR LS[PREVIOUS.ERROR] ;CLEAR PREVIOUS ERROR NUMBER
  
```

```

:33192 ; IN LS
:33193
U 1B62, B64A,15 :33194 MOV LS[IDC] TO WR[0] ;STORE MODULE CODE IN LS TO
U 1B63, 3E8C,15 :33195 MOV WR[0] TO LS[MODULE.NUM] ; INDICATE IDC MODULE
:33196
:33197
:33198
U 1B64, 37E0,15 :33199 MOV LS[#FFFFFFBFF] TO WR[0] ;MASK TO CHECK BIT <10>
U 1B65, 3E8A,15 :33200 MOV WR[0] TO LS[ERROR.MASK]
:33201
:33202 LOOP.T3C.1:
:33203 LOOP.T3C.2:
:33204
U 1B66, 8872,EC :33205 JSR [DIAG.CODE] ;INIT THE IDC AND FORCE IT
:33206 ; TO DIAG.IDLE WITH TIMEOUTS
:33207 ; DISABLED
:33208
U 1B67, B7E4,15 :33209 MOV LS[#10000080] TO WR[0] ;SET CRDY WITH TIMEOUTS
:33210 ; DISABLED
U 1B68, 17A0,15 :33211 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:33212
:33213 LOOP.T3C.3:
U 1B69, B640,15 :33214 MOV LS[#1] TO WR[0] ;SET WRO = 1
U 1B6A, BE82,15 :33215 MOV WR[0] TO LS[ERROR.NUMBER] ;SET UP ERROR NUMBER = 1 IN LS
:33216
:33217
U 1B6B, 2F80,15 :33218 CLR WR[0] ;CLR CRDY WITH TIMEOUTS ENABLED
U 1B6C, 17A0,15 :33219 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
U 1B6D, 37DE,95 :33220 MOV LS[#5A6C2] TO WR[1] ;SET UP TO LOOP 200ms
:33221 WAIT.T3C.200:
:33222 DEC WR[1],
:33223 DT(LONG)&SET.ALU.CC
U 1B6E, A102,B5 :33224 JMP.IF[NEQ] TO [WAIT.T3C.200] ;WAIT SOME MORE
U 1B6F, 09B6,E1 :33225
:33226
:33227 ;The IDC should timeout which will cause a JAM.
:33228 -----
:33229 -----
:33230 *****
:33231 * IDC *
:33232 *****
:33233 -----
:33234 =11111111
:33235
:33236 JAM:
:33237
:33238 UCMD/CLR.TIMEOUT,
:33239 BEN2/OPI.L, ;JAM TO THIS STATE ON POWER UP
:33240 NAD/PWR.FAIL ;OR TIMEOUT.
:33241 -----
:33242 =
:33243 =0**
:33244
:33245 TIMEOUT:
:33246

```

```

:33247 : UCMD/SET.CRDY, ;SIGNAL END OF TRANSFER.
:33248 : FUNC/SET.INT,
:33249 : NAD/IDLE
:33250 :-----
:33251
:33252 READ.CSR.T3C.1:
U 1B70, 9090,15 :33253 MISC [SET.PORT.I.0] ;SELECT THE IDC
U 1B71, 9780,15 :33254 PORT.READ PORT.ADRS[CSR] ;SET UP TO READ THE CSR
U 1B72, 3B32,15 :33255 MOV PORT TO LS[PORT0] ;READ THE DATA FROM CSR AND
:33256 ; SAVE IT IN LS
U 1B73, 3732,15 :33257 MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
U 1B74, 3654,95 :33258 MOV LS[BIT10] TO WR[1] ;SET UP EXPECTED DATA
U 1B75, 0869,3C :33259 JSR [CHECK.RESULT] ;CHECK THE DATA
U 1B76, 89B6,64 :33260 JMP [LOOP.T3C.1] ;LOOP HERE IF ERR & LOE IS SET
:33261
U 1B77, FF82,15 :33262 INC LS[ERROR.NUMBER] ;ERROR 2
U 1B78, B7E2,15 :33263 MOV LS[FFFFFFF7F] TO WR[0] ;SET UP THE ERROR MASK TO
U 1B79, 3E8A,15 :33264 MOV WR[0] TO LS[ERROR.MASK] ; CHECK BIT 7
U 1B7A, 3732,15 :33265 MOV LS[PORT0] TO WR[0] ;GET THE RECEIVED CSR DATA
U 1B7B, B64E,95 :33266 MOV LS[BIT7] TO WR[1] ;SET EXPECTED DATA
U 1B7C, 0869,3C :33267 JSR [CHECK.RESULT] ;CHECK THE DATA
U 1B7D, 89B6,64 :33268 JMP [LOOP.T3C.2] ;LOOP HERE IF ERR & LOE IS SET
:33269
U 1B7E, FF82,15 :33270 INC LS[ERROR.NUMBER] ;ERROR 3
U 1B7F, 37E0,15 :33271 MOV LS[FFFFFFBFF] TO WR[0] ;SET UP THE ERROR MASK TO CHECK
U 1B80, 3E8A,15 :33272 MOV WR[0] TO LS[ERROR.MASK] ; BIT 10
:33273
:33274 ;IDC SHOULD BE LOOPING IN THE IDC IDLE LOOP
:33275
U 1B81, 3720,15 :33276 MOV LS[SETCSR.MAINT] TO WR[0] ;GET DATA TO SET CSR MAINT BIT
:33277 ; CRDY = 0 AND MAINT = 1
U 1B82, 17A0,15 :33278 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE TO THE IDC CSR
:33279
U 1B83, B730,15 :33280 MOV LS[#1A] TO WR[0] ;SET UP A COUNTER TO NOP TO
:33281 ; ALLOW TIME TO BRANCH FROM
:33282 ; IDLE TO DIAG.IDLE
:33283
U 1B84, 2100,15 :33284 WAIT.BRANCH.T3C: ; IDC COMPLETES ITS OPERATION
:33285 DEC WR[0]
:33286 CMP WR[0] WITH LS[ZERO],
:33287 DT(LONG)&SET.ALU.CC
U 1B85, 4F9C,35 :33288 JMP.IF[NEQ] TO [WAIT.BRANCH.T3C]
:33289
:33290 ;CLEARING CSR <CRDY> WOULD HAVE CLEARED CSR <OPI> AND IF TIMEOUT HAS CLEARED,
:33291 ;CSR <OPI> WILL REMAIN CLEAR
:33292
:33293 READ.CSR.T3C.2:
U 1B87, 9090,15 :33294 MISC [SET.PORT.I.0] ;SELECT THE IDC
U 1B88, 9780,15 :33295 PORT.READ PORT.ADRS[CSR] ;SET UP TO READ THE CSR
U 1B89, 3B32,15 :33296 MOV PORT TO LS[PORT0] ;READ THE DATA FROM CSR AND
:33297 ; SAVE IT IN LS
U 1B8A, 3732,15 :33298 MOV LS[PORT0] TO WR[0] ;SET UP RECEIVED DATA
U 1B8B, 2F82,95 :33299 CLR WR[1] ;SET UP EXPECTED DATA
U 1B8C, 0869,3C :33300 JSR [CHECK.RESULT] ;CHECK THE DATA
U 1B8D, 89B6,94 :33301 JMP [LOOP.T3C.3] ;LOOP HERE IF ERR & LOE IS SET

```


:33339 .PAGE 'Test 3D TIMEOUT TEST 2 (M8388 IDC MODULE)'
:33340 :++

:33341 :
:33342 : Test Description:
:33343 :

:33344 : This test will check that the timeout counter times out within the
:33345 : specified amount of time which is 150 ms +50ms and -50ms after
:33346 : UCMD/RESET.TIMER has been issued with CSR <CRDY> clear.

:33347 :
:33348 : This test cannot be single-stepped because the timeout is generated
:33349 : from a one-shot.

:33350 :
:33351 : Assumptions:
:33352 :

:33353 : It is assumed that all previous tests have been run successfully.
:33354 :

:33355 : Test Steps:
:33356 :

- :33357 : 1. Set up error mask, error number and module number in LS
:33358 : (for error printouts) and clear the previous error number in LS.
:33359 : 2. Force the IDC to DIAG.IDLE.
:33360 : 3. Clear CSR <CRDY> with timeouts enabled to start the timer.
:33361 : 4. Let timer go 100ms.
:33362 : 5. Set up the CSR to branch to the correct IDC routine.

:33363 :
:33364 : *****
:33365 : * IDC *
:33366 : *****

:33367 :
:33368 : IDC1. FUNC/RESET.TIMER.
:33369 : IDC2. Send XFER REQ to flag the CPU.

- :33370 :
:33371 : 6. Wait for a XFER REQ from IDC.
:33372 : 7. Set up a loop to wait 200 ms.
:33373 : 8. Check CSR <OPI> (bit 10) = 1 to insure that the IDC timed out.
:33374 : 9. Increment the error number.
:33375 : 10. INIT the IDC and force the IDC to DIAG.IDLE.
:33376 : 11. Clear CSR <CRDY> to start the timer.
:33377 : 12. Let the timer go 100ms.
:33378 : 13. Set up the CSR to branch to the correct IDC routine.

:33379 :
:33380 :
:33381 : *****
:33382 : * IDC *
:33383 : *****

:33384 :
:33385 : IDC3. FUNC/RESET.TIMER.
:33386 : IDC4. Send XFER REQ to flag the CPU.

- :33387 :
:33388 : 14. Wait for a XFER REQ from the IDC.
:33389 : 15. Set up a loop to wait 100 ms.
:33390 : 16. Check CSR <OPI> (bit 10) = 0 to insure that IDC did not timeout.
:33391 : 17. Clear the IDC.
:33392 : 18. End test.
:33393 :

33394 : Errors:

Printout:

EXPECTED CSR <OPI>	RECEIVED CSR <OPI>
--------------------------	--------------------------

33401 : ERROR 1 - The IDC did not timeout within the specified time after
 33402 : FUNC/RESET.TIMER was executed and then CSR <CRDY> was
 33403 : cleared.

33404 : ERROR 2 - The IDC timed out too soon after a FUNC/RESET.TIMER was
 33405 : executed and with CSR <CRDY> clear.

33407 : Debug:

33409 : When CSR <CRDY> is cleared with URESET TIMER L = H, the one-shot
 33410 : is set up and should trigger 150ms (+50ms or -50ms) later if the
 33411 : set-up conditions remain unchanged. However, in this test after 100ms
 33412 : a URESET TIMER L is issued. In the microstate that the reset occurs in,
 33413 : UFUNC <3:0> = 0011 will be decoded to URESET TIMER L = L. This signal
 33414 : will reset the one-shot and the one-shot should trigger 150ms (+50 -50)
 33415 : after the reset.

33417 : --

33418 : *****
 33419 : T.3D:

U 1B9F, B65E,15	33420	MOV LS[BEGIN.TEST] TO WR[0]	: SET BIT 15 IN WR[0] FOR
	33421		: CONTROL AND STATUS WORD
U 1BA0, 3E80,15	33422	MOV WR[0] TO LS[CONTROL.STATUS]	: SET BIT 15 IN CONTROL AND
	33423		: STATUS WORD - BIT 15
	33424		: INDICATES START OF TEST TO
	33425		: CONSOLE
U 1BA1, 10E0,15	33426	MISC [SET.CP.ATTN]	: ISSUE CPU ATTN TO CONSOLE
	33427	WAIT.T3D.0:	
U 1BA2, 09BA,24	33428	JMP [WAIT.T3D.0]	: LOOP HERE WAITING FOR
	33429		: CONSOLE TO RESPOND
	33430		
U 1BA3, 65A0,15	33431	CLR LS[PREVIOUS.ERROR]	: CLEAR PREVIOUS ERROR NUMBER
	33432		: IN LS
	33433		
U 1BA4, B64A,15	33434	MOV LS[IDC] TO WR[0]	: STORE MODULE CODE IN LS TO
U 1BA5, 3E8C,15	33435	MOV WR[0] TO LS[MODULE.NUM]	: INDICATE IDC MODULE
	33436		
	33437		
U 1BA6, B640,15	33438	MOV LS[#1] TO WR[0]	: SET WRO = 1
U 1BA7, BE82,15	33439	MOV WR[0] TO LS[ERROR.NUMBER]	: SET UP ERROR NUMBER = 1 IN LS
	33440		
U 1BA8, 37E0,15	33441	MOV LS[#FFFFFFBFF] TO WR[0]	: SET UP THE ERROR MASK TO
U 1BA9, 3E8A,15	33442	MOV WR[0] TO LS[ERROR.MASK]	: CHECK BIT 10
	33443		
	33444		
	33445	LOOP.T3D.1:	
U 1BAA, 8872,EC	33446	JSR [DIAG.CODE]	: INIT THE IDC AND FORCE IT
	33447		: TO DIAG.IDLE
U 1BAB, B7E4,15	33448	MOV LS[#10000080] TO WR[0]	: SET CRDY WITH TIMEOUTS

```

:33449
U 1BAC, 17A0,15 :33450 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ; DISABLED
:33451 ;WRITE THE CSR
U 1BAD, 2F80,15 :33452 CLR WR[0]
U 1BAE, 17A0,15 :33453 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;CLR CRDY TO START TIMER AND
:33454 ;ENABLE TIMEOUTS
:33455
U 1BAF, B7DA,95 :33456 MOV LS[#28AEE] TO WR[1] ;SET UP TO WAIT 100ms
:33457 WAIT.T3D.100.1:
:33458 DEC WR[1],
:33459 DT(LONG)&SET.ALU.CC
U 1BB0, A102,B5 :33460 JMP.IF[NEQ] TO [WAIT.T3D.100.1] ;WAIT SOME MORE
:33461
:33462
:33463 ;SET UP THE CSR TO BRANCH TO THE IDC ROUTINE THAT WILL RESET.TIMER
:33464
U 1BB2, 370E,15 :33465 MOV LS[SETCSR.F7] TO WR[0] ;SET UP THE CSR TO BRANCH TO
:33466 ; THE CORRECT IDC ROUTINE
U 1BB3, C578,15 :33467 BIC LS[BIT28] TO WR[0] ;ENABLE TIMEOUTS
:33468
U 1BB4, 17A0,15 :33469 PORT.WRITE PORT.ADRS[CSR] WITH WR[0]
:33470
:33471 ;CONTINUE TO BRANCH
U 1BB5, 379E,15 :33472 MOV LS[SETCSR.R80] TO WR[0] ;CRDY REMAINS CLR
U 1BB6, 370A,95 :33473 MOV LS[SETCSR.F5] TO WR[1]
U 1BB7, AEC2,15 :33474 BIS WR[1] TO WR[0]
U 1BB8, C578,15 :33475 BIC LS[BIT28] TO WR[0] ;ENABLE TIMEOUTS
U 1BB9, 17A0,15 :33476 PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:33477 ;F<2:0>=110
:33478 ;FUNC/RESET.TIMER
:33479 -----
:33480 -----
:33481 *****
:33482 * IDC *
:33483 *****
:33484 -----
:33485 :018:
:33486 :=000
:33487 :DIAG.IDLE:
:33488 :
:33489 : BEN0/F0,
:33490 : BEN1/F1, ;BRANCH ON THE CSR F<2:0> BITS
:33491 : BEN2/F2,
:33492 : NAD/DIAG.IDLE
:33493 -----
:33494 :01F:
:33495 :=111
:33496 :
:33497 : BEN0/CRDY.L,
:33498 : BEN2/MAINT, ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:33499 : NAD/TEST.FUNC7 ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:33500 -----
:33501 :079:
:33502 :=0*1
:33503

```

```

:33504 : NAD/GROUP1
:33505 : -----
:33506 : 182:
:33507 : =0**
:33508 : GROUP1:
:33509 :
:33510 : BEN2/R80.FORMAT.H, ; WHEN THE R80 FORMAT BIT SETS..BRANCH
:33511 : NAD/GROUP1 ; TO NEXT WORD AND THEN BRANCH ON THE
:33512 : ; FUNCTION BITS
:33513 : -----
:33514 : 186:
:33515 : =1**
:33516 :
:33517 : BEN0/F0,
:33518 : BEN1/F1, ; BRANCH ON THE CSR F<2:0> BITS
:33519 : BEN2/F2,
:33520 : NAD/GROUP1.F0
:33521 : -----
:33522 : 02D:
:33523 : =101
:33524 : RESET.TIMER:
:33525 :
:33526 : FUNC/RESET.TIMER, ; RESET THE TIMEOUT COUNTER...BE SURE CRDY
:33527 : NAD/XFER.REQ ; IS SET
:33528 : -----
:33529 : 1D8:
:33530 : XFER.REQ:
:33531 : UCMD/SET.XFER.RQST, ; SEND XFER REQ TO THE CPU
:33532 : NAD/DIAG.WAIT
:33533 : -----
:33534 : 1D7:
:33535 : DIAG.WAIT:
:33536 :
:33537 : BEN1/XFER.REQ, ; WAIT FOR THE CPU TO ISSUE XFER GRANT
:33538 : NAD/DIAG.WAIT1 ; BEFORE RETURNING TO DIAG.IDLE
:33539 : -----
:33540 : 159:
:33541 : =*0*
:33542 : DIAG.WAIT1:
:33543 :
:33544 : NAD/DIAG.IDLE ; RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:33545 : ; COMMAND FROM THE CPU
:33546 : -----
:33547 : 15B:
:33548 : =*1*
:33549 : DIAG.WAIT2:
:33550 :
:33551 : NAD/DIAG.WAIT ; XFER GRANT HAS NOT BEEN ISSUED....
:33552 : ; WAIT SOME MORE
:33553 : -----
:33554 : WAIT.IDC.T3D.1:
:33555 : SKIP.IF[NO.FAST.INTERRUPT] ; WAIT FOR XFER REQ FROM IDC
:33556 : JMP [IDC.DONE.T3D.1] ; XFER REQ SET
:33557 : JMP [WAIT.IDC.T3D.1] ; WAIT SOME MORE
:33558 :

```

.....

.....

```

U 1BD9, AEC2,15 :33614      BIS WR[1] TO WR[0]
U 1BDA, C578,15 :33615      BIC LS[BIT28] TO WR[0]      ;ENABLE TIMEOUTS
U 1BDB, 17A0,15 :33616      PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
:33617      ;F<2:0>=101
:33618      ;RESET.TIMER
:33619      -----
:33620      -----
:33621      *****
:33622      * IDC *
:33623      *****
:33624      -----
:33625      018:
:33626      =000
:33627      DIAG.IDLE:
:33628
:33629      BEN0/F0,
:33630      BEN1/F1,      ;BRANCH ON THE CSR F<2:0> BITS
:33631      BEN2/F2,
:33632      NAD/DIAG.IDLE
:33633      -----
:33634      01F:
:33635      =111
:33636
:33637      BEN0/CRDY.L,
:33638      BEN2/MAINT,      ;BRANCH TO THE CORRECT IDC MICROCODE ROUTINE
:33639      NAD/TEST.FUNC7      ; DEPENDENT ON CSR <CRDY> AND <MAINT>
:33640      -----
:33641      079:
:33642      =0*1
:33643
:33644      NAD/GROUP1
:33645      -----
:33646      182:
:33647      =0**
:33648      GROUP1:
:33649
:33650      BEN2/R80.FORMAT.H,      ;WHEN THE R80 FORMAT BIT SETS..BRANCH
:33651      NAD/GROUP1      ; TO NEXT WORD AND THEN BRANCH ON THE
:33652      ; FUNCTION BITS
:33653      -----
:33654      186:
:33655      =1**
:33656
:33657      BEN0/F0,
:33658      BEN1/F1,      ;BRANCH ON THE CSR F<2:0> BITS
:33659      BEN2/F2,
:33660      NAD/GROUP1.F0
:33661      -----
:33662      02D:
:33663      =101
:33664      RESET.TIMER:
:33665
:33666      FUNC/RESET.TIMER,      ;RESET THE TIMEOUT COUNTER...BE SURE CRDY
:33667      NAD/XFER.REQ      ; IS SET
:33668      -----
  
```

```

:33669 :1D8:
:33670 :XFER.REQ:
:33671 :   UCMD/SET.XFER.RQST,      ;SEND XFER REQ TO THE CPU
:33672 :   NAD/DIAG.WAIT
:33673 -----
:33674 :1D7:
:33675 :DIAG.WAIT:
:33676 :
:33677 :   BEN1/XFER.REQ,      ;WAIT FOR THE CPU TO ISSUE XFER GRANT
:33678 :   NAD/DIAG.WAIT1      ; BEFORE RETURNING TO DIAG.IDLE
:33679 -----
:33680 :159:
:33681 :=*0*
:33682 :DIAG.WAIT1:
:33683 :
:33684 :   NAD/DIAG.IDLE      ;RETURN TO DIAG.IDLE TO WAIT FOR NEXT COMMAND
:33685 :                       ; COMMAND FROM THE CPU
:33686 -----
:33687 :15B:
:33688 :=*1*
:33689 :DIAG.WAIT2:
:33690 :
:33691 :   NAD/DIAG.WAIT      ;XFER GRANT HAS NOT BEEN ISSUED....
:33692 :                       ; WAIT SOME MORE
:33693 -----
:33694 :WAIT.IDC.T3D.2:
U 1BDC, DB00,1A :33695 :   SKIP.IF[NO.FAST.INTERRUPT]      ;WAIT FOR XFER REQ FROM IDC
U 1BDD, 09BD,F4 :33696 :   JMP [IDC.DONE.T3D.2]            ;XFER REQ SET
U 1BDE, 09BD,C4 :33697 :   JMP [WAIT.IDC.T3D.2]            ;WAIT SOME MORE
:33698 :
:33699 :IDC.DONE.T3D.2:
U 1BDF, B700,15 :33700 :   MOV LS[SETCSR.F0] TO WR[0]      ;CLEAR CSR F<2:0>
U 1BE0, C578,15 :33701 :   BIC LS[BIT28] TO WR[0]          ;ENABLE TIMEOUTS
U 1BE1, 17A0,15 :33702 :   PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE THE CSR
U 1BE2, 90FA,15 :33703 :   MISC2 [TRANSFER.GRANT]          ;SEND GRANT TO CLR THE XFER REQ
:33704 :
U 1BE3, B7DA,95 :33705 :   MOV LS[#28AEE] TO WR[1]         ;SET UP TO WAIT 100ms
:33706 :WAIT.T3D.100:
:33707 :   DEC WR[1],
U 1BE4, A102,B5 :33708 :   DT(LONG)&SET.ALU.CC
U 1BE5, 89BE,41 :33709 :   JMP.IF[NEQ] TO [WAIT.T3D.100]    ;WAIT SOME MORE
:33710 :
:33711 :READ.CSR.T3D.2:
U 1BE6, 9090,15 :33712 :   MISC [SET.PORT.1.0]             ;SELECT THE IDC
U 1BE7, 9780,15 :33713 :   PORT.READ PORT.ADRS[CSR]        ;SET UP TO READ THE CSR
U 1BE8, 3B32,15 :33714 :   MOV PORT TO LS[PORT0]           ;READ THE DATA FROM CSR AND
:33715 :   ; SAVE IT IN LS
U 1BE9, 3732,15 :33716 :   MOV LS[PORT0] TO WR[0]          ;SET UP RECEIVED DATA
U 1BEA, 2F82,95 :33717 :   CLR WR[1]                       ;SET UP EXPECTED DATA
U 1BEB, 0869,3C :33718 :   JSR [CHECK.RESULT]              ;CHECK THE DATA
U 1BEC, 89BC,C4 :33719 :   JMP [LOOP.T3D.2]                ;LOOP HERE IF ERR & LOE IS SET
:33720 :
:33721 :
U 1BED, 8875,3C :33722 :   JSR [CLEANUP]                   ;CLEANUP THE CONDITIONS SET
:33723 :                                   ; BY THIS TEST
  
```


MICRO2 1M(01) Cross Reference Listing - Field Names and Defined Values
15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module
Fiche 4 Frame K5
Sequence 680
Rev 01.40
Page 674

[illegible]

MICRO2 1M(01) Cross Reference Listing - Macro Names
15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module
Fiche 4 Frame L5 Sequence 681
Rev 01.40 Page 675

LO

M 5
MICRO2 1M(01) Cross Reference Lis15-MAR-83 Fiche 4 Frame M5 Sequence 682
15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40 Page 676
Cross Reference Listing - Expression Names

[illegible]

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99

Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
C 01A8	3075:	3076:	3077:	3078:	3080:	3081:	3082:	3083:
C 01B0	3086:	3087:	3088:	3089:	3091:	3092:	3093:	3094:
C 01B8	3096:	3097:	3098:	3099:	3101:	3102:	3103:	3104:
C 01C0	3107:	3108:	3109:	3110:	3112:	3113:	3114:	3115:
C 01C8	3117:	3118:	3119:	3120:	3122:	3123:	3124:	3125:
C 01D0	3128:	3129:	3130:	3131:	3133:	3134:	3135:	3136:
C 01D8	3138:	3139:	3140:	3141:	3143:	3144:	3145:	3146:
C 01E0	3149:	3150:	3151:	3152:	3154:	3155:	3156:	3157:
C 01E8	3159:	3160:	3161:	3162:	3164:	3165:	3166:	3167:
C 01F0	3170:	3171:	3172:	3173:	3175:	3176:	3177:	3178:
C 01F8	3180:	3181:	3182:	3183:	3185:	3186:	3187:	3188:

:Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 0000	4788:	4669:	4670:	4671:	4672:	4673:	4674:	4675:
U 0008	4676:	4677:	4678:	4679:	4680:	4681:	4682:	4683:
U 0010	4684:	4685:	4686:	4687:	4688:	4689:	4690:	4691:
U 0018	4692:	4693:	4694:	4695:	4696:	4697:	4698:	4699:
U 0020	4700:	4701:	4702:	4703:	4704:	4705:	4706:	4707:
U 0028	4708:	4709:	4710:	4711:	4712:	4713:	4714:	4715:
U 0030	4716:	4717:	4718:	4719:	4720:	4721:	4722:	4723:
U 0038	4724:	4725:	4726:	4727:	4728:	4729:	4730:	4732:
U 0040	4734:	4736:	4738:	4740:	4742:	4744:	4746:	4748:
U 0048	4750:	4752:	4754:	4756:	4758:	4760:	4762:	4764:
U 0050	4797:	4799:	4801:	4804:	4807:	4811:	4813:	4815:
U 0058	4817:	4819:	4821:	4823:	4825:	4827:	4829:	4831:
U 0060	4833:	4835:	4837:	4839:	4841:	4843:	4844:	4845:
U 0068	4846:	4847:	4848:	4849:	4850:	4851:	4852:	4853:
U 0070	4883:	4885:	4887:	4888:	4890:	4891:	4893:	4894:
U 0078	4896:	4897:	4899:	4900:	4902:	4903:	4905:	4906:
U 0080	4908:	4909:	4911:	4912:	4914:	4915:	4917:	4918:
U 0088	4920:	4921:	4923:	4924:	4926:	4927:	4929:	4930:
U 0090	4932:	4933:	4935:	4936:	4938:	4939:	4941:	4942:
U 0098	4944:	4945:	4947:	4948:	4950:	4951:	4953:	4954:
U 00A0	4956:	4957:	4959:	4960:	4962:	4963:	4965:	4966:
U 00A8	4968:	4969:	4971:	4972:	4974:	4975:	4977:	4978:
U 00B0	4980:	4981:	4983:	4984:	4986:	4987:	4989:	4990:
U 00B8	4992:	4993:	4995:	4996:	4998:	4999:	5001:	5002:
U 00C0	5004:	5005:	5007:	5008:	5010:	5011:	5013:	5014:
U 00C8	5016:	5017:	5019:	5020:	5022:	5023:	5025:	5026:
U 00D0	5028:	5029:	5031:	5032:	5034:	5035:	5037:	5038:
U 00D8	5040:	5041:	5043:	5044:	5046:	5047:	5049:	5050:
U 00E0	5052:	5053:	5055:	5056:	5058:	5059:	5061:	5062:
U 00E8	5064:	5065:	5067:	5068:	5070:	5071:	5073:	5074:
U 00F0	5076:	5077:	5079:	5080:	5082:	5083:	5085:	5086:
U 00F8	5088:	5089:	5091:	5092:	5094:	5095:	5097:	5098:
U 0100	5100:	5101:	5103:	5104:	5106:	5107:	5109:	5110:
U 0108	5112:	5113:	5115:	5116:	5118:	5119:	5121:	5122:
U 0110	5124:	5125:	5127:	5128:	5130:	5131:	5133:	5134:
U 0118	5136:	5137:	5139:	5140:	5142:	5143:	5145:	5146:
U 0120	5148:	5149:	5151:	5152:	5154:	5155:	5157:	5158:
U 0128	5160:	5161:	5163:	5164:	5166:	5167:	5169:	5170:
U 0130	5172:	5173:	5175:	5176:	5178:	5179:	5181:	5182:
U 0138	5184:	5185:	5187:	5188:	5190:	5191:	5193:	5194:
U 0140	5196:	5197:	5199:	5200:	5202:	5203:	5205:	5206:
U 0148	5208:	5209:	5211:	5212:	5214:	5215:	5217:	5218:
U 0150	5220:	5221:	5223:	5224:	5226:	5227:	5229:	5230:
U 0158	5232:	5233:	5235:	5236:	5238:	5239:	5241:	5242:
U 0160	5244:	5245:	5254:	5256:	5258:	5259:	5261:	5262:
U 0168	5264:	5265:	5267:	5268:	5270:	5271:	5273:	5274:
U 0170	5276:	5277:	5279:	5280:	5282:	5283:	5285:	5286:
U 0178	5288:	5289:	5291:	5292:	5294:	5295:	5297:	5298:
U 0180	5300:	5301:	5303:	5304:	5306:	5307:	5309:	5310:
U 0188	5312:	5313:	5315:	5316:	5318:	5319:	5321:	5322:
U 0190	5324:	5325:	5327:	5328:	5330:	5331:	5333:	5334:
U 0198	5336:	5337:	5339:	5340:	5342:	5343:	5345:	5346:
U 01A0	5348:	5349:	5351:	5352:	5354:	5355:	5357:	5358:

:Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F	:L
U 01A8	5360:	5361:	5363:	5364:	5366:	5367:	5369:	5370:	U
U 01B0	5372:	5373:	5375:	5376:	5378:	5379:	5381:	5382:	U
U 01B8	5384:	5385:	5387:	5388:	5390:	5391:	5393:	5394:	U
U 01C0	5396:	5397:	5399:	5400:	5402:	5403:	5405:	5406:	U
U 01C8	5408:	5409:	5411:	5412:	5414:	5415:	5417:	5418:	U
U 01D0	5420:	5421:	5423:	5424:	5426:	5427:	5429:	5430:	U
U 01D8	5432:	5433:	5435:	5436:	5438:	5439:	5441:	5442:	U
U 01E0	5444:	5445:	5447:	5448:	5450:	5451:	5453:	5454:	U
U 01E8	5456:	5457:	5459:	5460:	5462:	5463:	5465:	5466:	U
U 01F0	5468:	5469:	5471:	5472:	5474:	5475:	5477:	5478:	U
U 01F8	5480:	5481:	5483:	5484:	5486:	5487:	5489:	5490:	U
U 0200	5492:	5493:	5495:	5496:	5498:	5499:	5501:	5502:	U
U 0208	5504:	5505:	5507:	5508:	5510:	5511:	5513:	5514:	U
U 0210	5516:	5517:	5519:	5520:	5522:	5523:	5525:	5526:	U
U 0218	5528:	5529:	5531:	5532:	5534:	5535:	5537:	5538:	U
U 0220	5540:	5541:	5543:	5544:	5546:	5547:	5549:	5550:	U
U 0228	5552:	5553:	5555:	5556:	5558:	5559:	5561:	5562:	U
U 0230	5564:	5565:	5567:	5568:	5570:	5571:	5573:	5574:	U
U 0238	5576:	5577:	5579:	5580:	5582:	5583:	5585:	5586:	U
U 0240	5588:	5589:	5591:	5592:	5594:	5595:	5597:	5598:	U
U 0248	5600:	5601:	5603:	5604:	5606:	5607:	5609:	5610:	U
U 0250	5612:	5613:	5615:	5616:					U
U 0258 - 05FF Unused									U
U 0600	5639:	5640:	5642:	5645:	5646:	5647:	5661:	5662:	U
U 0608	5723:	5725:	5727:	5729:	5730:	5785:	5786:	5788:	U
U 0610	5790:	5792:	5795:	5797:	5798:	5799:	5803:	5804:	U
U 0618	5805:	5809:	5810:	5811:	5813:	5816:	5817:	5872:	U
U 0620	5874:	5875:	5878:	5880:	5884:	5886:	5890:	5896:	U
U 0628	5898:	5899:	5905:	5907:	5909:	5910:	5911:	5917:	U
U 0630	5918:	5919:	5920:	5921:	5922:	5923:	5926:	5928:	U
U 0638	5929:	5933:	5983:	5985:	5987:	5989:	5992:	5993:	U
U 0640	5995:	5996:	6051:	6053:	6054:	6057:	6059:	6063:	U
U 0648	6065:	6069:	6075:	6077:	6078:	6084:	6085:	6086:	U
U 0650	6087:	6088:	6089:	6092:	6142:	6144:	6147:	6149:	U
U 0658	6152:	6153:	6155:	6156:	6204:	6209:	6211:	6213:	U
U 0660	6214:	6215:	6265:	6267:	6270:	6274:	6276:	6277:	U
U 0668	6316:	6317:	6318:	6321:	6322:	6323:	6326:	6327:	U
U 0670	6328:	6331:	6332:	6333:	6336:	6337:	6341:	6342:	U
U 0678	6343:	6382:	6383:	6384:	6388:	6389:	6390:	6394:	U
U 0680	6395:	6396:	6400:	6401:	6405:	6406:	6410:	6413:	U
U 0688	6415:	6462:	6463:	6464:	6465:	6466:	6513:	6514:	U
U 0690	6515:	6516:	6517:	6585:	6587:	6590:	6591:	6593:	U
U 0698	6595:	6596:	6597:	6599:	6603:	6605:	6607:	6611:	U
U 06A0	6612:	6615:	6616:	6618:	6620:	6621:	6623:	6625:	U
U 06A8	6630:	6631:	6633:	6635:	6636:	6642:	6643:	6646:	U
U 06B0	6647:	6648:	6653:	6654:	6656:	6657:	6659:	6661:	U
U 06B8	6662:	6664:	6666:	6668:	6670:	6672:	6679:	6680:	U
U 06C0	6682:	6685:	6698:	6699:	6701:	6702:	6703:	6704:	U
U 06C8	6705:	6706:	6707:	6708:	6709:	6713:	6716:	6717:	U
U 06D0	6718:	6719:	6720:	6721:	6722:	6723:	6724:	6725:	U
U 06D8	6726:	6727:	6728:	6729:	6730:	6731:	6734:	6737:	U
U 06E0	6740:	6742:	6743:	6744:	6745:	6748:	6751:	6754:	U
U 06E8	6756:	6757:	6758:	6760:	6761:	6762:	6764:	6766:	U

ZZ-ENKCF-1.4 ;		MICRO2 1M(01)		Location / Line Num15-MAR-83		Fiche 4 Frame E6		Sequence 687		Page 681		ZZ
: ENKCF.MCR				15-MAR-83 11:46:22		ENKCF Micro-diagnostic for IDC module		Rev 01.40				:
:												:
:Location		0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F			
U 06F0	6767:	6768:	6769:	6772:	6773:	6776:	6778:	6779:				
U 06F8 - 06FF	Unused											
U 0700	6832:	6833:	6836:	6837:	6839:	6841:	6842:	6845:				Us
U 0708	6846:	6848:	6849:	6850:	6851:	6854:	6855:	6857:				Re
U 0710	6859:	6860:	6862:	6863:	6867:	6868:	6869:	6871:				To
U 0718	6873:	6875:	6880:	6881:	6884:	6885:	6887:	6889:				To
U 0720	6892:	6893:	6895:	6896:	6898:	6900:	6901:	6903:				Hi
U 0728	6905:	6906:	6908:	6912:	6914:	6916:	6966:	6967:				
U 0730	6968:	6969:	6970:	6972:	6973:	6975:	6980:	6982:				
U 0738	6983:	6984:	6985:	6987:	6991:	7040:	7041:	7042:				
U 0740	7043:	7045:	7046:	7047:	7049:	7050:	7201:	7202:				
U 0748	7203:	7208:	7209:	7210:	7211:	7212:	7213:	7214:				
U 0750	7215:	7216:	7217:	7260:	7262:	7263:	7264:	7322:				
U 0758	7324:	7328:	7330:	7332:	7335:	7336:	7337:	7339:				
U 0760	7340:	7342:	7346:	7347:	7348:	7349:	7350:	7351:				
U 0768	7353:	7354:	7355:	7356:	7357:	7358:	7360:	7361:				
U 0770	7362:	7363:	7364:	7365:	7367:	7368:	7369:	7370:				
U 0778	7371:	7372:	7374:	7375:	7376:	7377:	7378:	7379:				
U 0780	7381:	7382:	7383:	7384:	7385:	7386:	7461:	7463:				
U 0788	7467:	7469:	7472:	7473:	7474:	7476:	7477:	7478:				
U 0790	7479:	7485:	7486:	7489:	7490:	7492:	7493:	7494:				
U 0798	7495:	7496:	7498:	7500:	7503:	7504:	7507:	7508:				
U 07A0	7511:	7512:	7514:	7515:	7516:	7517:	7518:	7520:				
U 07A8	7522:	7525:	7526:	7529:	7530:	7531:	7533:	7536:				
U 07B0	7538:	7539:	7541:	7542:	7543:	7544:	7545:	7546:				
U 07B8	7548:	7552:	7553:	7554:	7555:	7557:	7671:	7673:				
U 07C0	7677:	7679:	7682:	7685:	7686:	7687:	7690:	7691:				
U 07C8	7693:	7694:	7697:	7698:	7699:	7700:	7701:	7702:				
U 07D0	7703:	7704:	7705:	7706:	7710:	7711:	7712:	7713:				
U 07D8	7714:	7715:	7716:	7717:	7718:	7719:	7720:	7721:				
U 07E0	7723:	7724:	7725:	7726:	7849:	7851:	7855:	7857:				
U 07E8	7860:	7863:	7864:	7866:	7867:	7869:	7870:	7873:				
U 07F0	7875:	7876:	7879:	7880:	7881:	7882:	7883:	7886:				
U 07F8	7887:	7888:	7890:	7891:	7892:	7893:	7897:	7898:				
U 0800	7899:	7900:	7901:	7904:	7905:	7906:	7908:	7909:				
U 0808	7910:	7911:	7915:	7916:	7917:	7918:	7919:	7922:				
U 0810	7923:	7924:	7926:	7927:	7928:	7929:	7933:	7934:				
U 0818	7935:	7936:	7937:	7940:	7941:	7942:	7944:	7945:				
U 0820	7946:	7947:	7951:	7952:	7953:	7954:	7955:	7958:				
U 0828	7959:	7960:	7962:	7963:	7964:	7965:	7969:	7970:				
U 0830	7971:	7972:	7973:	7976:	7977:	7978:	7980:	7981:				
U 0838	7982:	7983:	7986:	7989:	7990:	7991:	7992:	7993:				
U 0840	7994:	7995:	7996:	7998:	7999:	8000:	8001:	8002:				
U 0848	8146:	8148:	8152:	8154:	8157:	8160:	8161:	8163:				
U 0850	8164:	8166:	8167:	8170:	8172:	8173:	8175:	8178:				
U 0858	8181:	8182:	8183:	8185:	8186:	8187:	8188:	8191:				
U 0860	8194:	8197:	8198:	8199:	8201:	8202:	8203:	8204:				
U 0868	8207:	8210:	8213:	8214:	8215:	8217:	8218:	8219:				
U 0870	8220:	8223:	8226:	8229:	8230:	8231:	8233:	8234:				
U 0878	8235:	8236:	8239:	8242:	8245:	8246:	8247:	8249:				
U 0880	8250:	8251:	8252:	8255:	8256:	8257:	8260:	8261:				
U 0888	8262:	8263:	8264:	8265:	8266:	8267:	8268:	8269:				
U 0890	8271:	8272:	8273:	8274:	8277:	8278:	8279:	8282:				

:Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 0898	8283:	8284:	8285:	8286:	8287:	8288:	8289:	8290:
U 08A0	8291:	8293:	8294:	8295:	8296:	8298:	8299:	8300:
U 08A8	8301:	8304:	8477:	8479:	8483:	8485:	8487:	8489:
U 08B0	8490:	8493:	8494:	8496:	8498:	8501:	8503:	8504:
U 08B8	8506:	8507:	8509:	8510:	8512:	8514:	8516:	8517:
U 08C0	8554:	8555:	8556:	8557:	8558:	8559:	8560:	8561:
U 08C8	8562:	8564:	8565:	8567:	8568:	8569:	8570:	8571:
U 08D0	8572:	8573:	8576:	8577:	8578:	8581:	8582:	8583:
U 08D8	8584:	8585:	8586:	8587:	8588:	8589:	8590:	8593:
U 08E0	8594:	8595:	8598:	8599:	8600:	8601:	8602:	8603:
U 08E8	8604:	8605:	8606:	8607:	8610:	8611:	8612:	8615:
U 08F0	8616:	8617:	8618:	8619:	8620:	8621:	8622:	8623:
U 08F8	8624:	8626:	8725:	8727:	8731:	8733:	8736:	8739:
U 0900	8740:	8743:	8744:	8746:	8747:	8752:	8756:	8757:
U 0908	8759:	8761:	8762:	8763:	8796:	8797:	8798:	8800:
U 0910	8801:	8802:	8803:	8804:	8805:	8806:	8808:	8811:
U 0918	8815:	8816:	8818:	8820:	8821:	8822:	8854:	8855:
U 0920	8856:	8858:	8859:	8860:	8861:	8862:	8864:	8865:
U 0928	8867:	8869:	8873:	8874:	8876:	8878:	8879:	8880:
U 0930	8913:	8914:	8916:	8917:	8918:	8919:	8920:	8922:
U 0938	8923:	8925:	9026:	9028:	9032:	9034:	9037:	9040:
U 0940	9041:	9042:	9045:	9046:	9048:	9049:	9050:	9052:
U 0948	9055:	9056:	9057:	9058:	9090:	9091:	9092:	9093:
U 0950	9094:	9095:	9097:	9099:	9100:	9101:	9102:	9105:
U 0958	9106:	9107:	9108:	9110:	9111:	9112:	9114:	9116:
U 0960	9117:	9118:	9119:	9122:	9123:	9124:	9125:	9128:
U 0968	9129:	9130:	9131:	9134:	9246:	9248:	9252:	9254:
U 0970	9257:	9260:	9261:	9265:	9266:	9268:	9270:	9273:
U 0978	9274:	9276:	9279:	9280:	9282:	9283:	9284:	9286:
U 0980	9287:	9288:	9289:	9291:	9293:	9295:	9296:	9297:
U 0988	9300:	9302:	9303:	9304:	9305:	9341:	9342:	9343:
U 0990	9345:	9346:	9347:	9351:	9352:	9353:	9355:	9356:
U 0998	9357:	9358:	9361:	9362:	9364:	9502:	9504:	9508:
U 09A0	9510:	9513:	9516:	9517:	9520:	9521:	9523:	9524:
U 09A8	9526:	9529:	9530:	9532:	9535:	9574:	9575:	9576:
U 09B0	9578:	9579:	9580:	9582:	9583:	9584:	9586:	9587:
U 09B8	9588:	9589:	9591:	9594:	9595:	9597:	9598:	9599:
U 09C0	9600:	9602:	9603:	9649:	9650:	9651:	9653:	9654:
U 09C8	9655:	9658:	9659:	9660:	9662:	9663:	9664:	9665:
U 09D0	9667:	9670:	9671:	9673:	9674:	9676:	9677:	9729:
U 09D8	9730:	9731:	9733:	9734:	9735:	9737:	9738:	9739:
U 09E0	9741:	9742:	9743:	9744:	9746:	9802:	9804:	9808:
U 09E8	9810:	9813:	9816:	9817:	9820:	9821:	9823:	9824:
U 09F0	9828:	9829:	9830:	9832:	9834:	9836:	9837:	9842:
U 09F8	9845:	9846:	9849:	9850:	9851:	9853:	9854:	9855:
U 0A00	9856:	9858:	9960:	9962:	9966:	9968:	9971:	9974:
U 0A08	9975:	9978:	9979:	9981:	9982:	9984:	9985:	9987:
U 0A10	9990:	9992:	9993:	9997:	9998:	10000:	10001:	10002:
U 0A18	10003:	10004:	10077:	10078:	10079:	10082:	10083:	10084:
U 0A20	10087:	10088:	10089:	10091:	10093:	10096:	10097:	10098:
U 0A28	10101:	10102:	10103:	10105:	10106:	10107:	10108:	10109:
U 0A30	10111:	10112:	10113:	10116:	10117:	10119:	10226:	10228:
U 0A38	10232:	10234:	10237:	10240:	10241:	10244:	10245:	10247:

Us
Rel

To
To
Hi

EN

Pa
Pa

:Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U OBE8	12147:	12148:	12149:	12150:	12151:	12152:	12194:	12197:
U OBF0	12198:	12200:	12201:	12203:	12204:	12205:	12206:	12207:
U OBF8	12211:	12212:	12215:	12216:	12218:	12220:	12222:	12223:
U OC00	12226:	12227:	12229:	12230:	12232:	12233:	12234:	12235:
U OC08	12236:	12240:	12241:	12243:	12244:	12245:	12247:	12249:
U OC10	12252:	12253:	12256:	12257:	12259:	12260:	12261:	12262:
U OC18	12263:	12268:	12269:	12270:	12271:	12272:	12276:	12491:
U OC20	12493:	12497:	12499:	12502:	12505:	12506:	12510:	12512:
U OC28	12513:	12515:	12516:	12520:	12522:	12523:	12525:	12529:
U OC30	12533:	12534:	12535:	12537:	12538:	12540:	12542:	12543:
U OC38	12546:	12547:	12550:	12551:	12552:	12555:	12556:	12559:
U OC40	12560:	12563:	12564:	12567:	12570:	12572:	12573:	12574:
U OC48	12575:	12576:	12643:	12644:	12645:	12648:	12649:	12650:
U OC50	12652:	12653:	12654:	12655:	12715:	12716:	12717:	12719:
U OC58	12720:	12721:	12725:	12726:	12728:	12730:	12731:	12732:
U OC60	12733:	12734:	12735:	12736:	12777:	12780:	12781:	12783:
U OC68	12784:	12787:	12788:	12789:	12790:	12791:	12794:	12795:
U OC70	12798:	12799:	12801:	12803:	12805:	12806:	12809:	12810:
U OC78	12813:	12814:	12816:	12817:	12818:	12819:	12820:	12824:
U OC80	12825:	12827:	12828:	12830:	12832:	12835:	12836:	12838:
U OC88	12839:	12840:	12842:	12843:	12844:	12845:	12846:	12850:
U OC90	12851:	12852:	12853:	12854:	12857:	13107:	13109:	13113:
U OC98	13115:	13118:	13121:	13122:	13126:	13128:	13129:	13131:
U OCA0	13132:	13136:	13138:	13139:	13141:	13145:	13149:	13151:
U OCA8	13152:	13153:	13154:	13156:	13160:	13162:	13163:	13164:
U OCB0	13165:	13166:	13234:	13235:	13236:	13239:	13240:	13241:
U OCB8	13244:	13245:	13247:	13249:	13250:	13251:	13253:	13254:
U OCC0	13298:	13301:	13302:	13304:	13305:	13307:	13308:	13309:
U OCC8	13310:	13311:	13313:	13314:	13318:	13319:	13321:	13323:
U OCD0	13325:	13327:	13330:	13331:	13334:	13335:	13337:	13338:
U OCD8	13339:	13340:	13341:	13345:	13346:	13348:	13349:	13351:
U OCE0	13353:	13354:	13357:	13358:	13360:	13361:	13362:	13365:
U OCE8	13366:	13367:	13368:	13369:	13371:	13372:	13374:	13375:
U OCF0	13378:	13384:	13385:	13435:	13438:	13439:	13441:	13442:
U OCF8	13445:	13446:	13447:	13448:	13449:	13450:	13453:	13454:
U OD00	13455:	13456:	13457:	13458:	13461:	13462:	13463:	13464:
U OD08	13465:	13470:	13712:	13714:	13718:	13720:	13723:	13726:
U OD10	13727:	13731:	13733:	13734:	13736:	13737:	13740:	13742:
U OD18	13743:	13745:	13749:	13753:	13754:	13755:	13757:	13758:
U OD20	13760:	13762:	13763:	13766:	13767:	13770:	13771:	13773:
U OD28	13776:	13777:	13780:	13781:	13784:	13785:	13788:	13791:
U OD30	13793:	13794:	13795:	13796:	13797:	13865:	13866:	13867:
U OD38	13870:	13871:	13872:	13876:	13877:	13879:	13881:	13882:
U OD40	13883:	13885:	13886:	13929:	13932:	13933:	13937:	13938:
U OD48	13941:	13942:	13943:	13944:	13945:	13948:	13949:	13953:
U OD50	13954:	13955:	13957:	13959:	13961:	13963:	13966:	13967:
U OD58	13970:	13971:	13973:	13974:	13975:	13976:	13977:	13981:
U OD60	13982:	13984:	13985:	13987:	13989:	13991:	13994:	13995:
U OD68	13998:	13999:	14000:	14003:	14004:	14005:	14006:	14007:
U OD70	14009:	14010:	14013:	14014:	14018:	14022:	14023:	14073:
U OD78	14076:	14077:	14081:	14082:	14085:	14086:	14087:	14088:
U OD80	14089:	14090:	14094:	14095:	14096:	14097:	14098:	14101:
U OD88	14102:	14103:	14104:	14105:	14109:	14373:	14375:	14379:

ZZ-ENKCF-1.4 ;
: ENKCF.MCR
:

Location / Line Num15-MAR-83
MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module
Location / Line Number Index

Fiche 4 Frame 16

Sequence 691
Rev 01.40 Page 685

ZZ

Fi

:Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U OD90	14381:	14384:	14387:	14388:	14392:	14394:	14395:	14397:
U OD98	14398:	14401:	14403:	14404:	14406:	14411:	14415:	14417:
U ODA0	14418:	14419:	14420:	14422:	14426:	14428:	14430:	14431:
U ODA8	14432:	14433:	14499:	14500:	14501:	14504:	14505:	14506:
U ODB0	14508:	14509:	14510:	14511:	14570:	14571:	14572:	14574:
U ODB8	14575:	14576:	14580:	14581:	14583:	14585:	14586:	14587:
U ODC0	14589:	14590:	14633:	14636:	14637:	14642:	14643:	14646:
U ODC8	14647:	14648:	14649:	14650:	14654:	14655:	14657:	14658:
U ODD0	14660:	14662:	14664:	14666:	14669:	14670:	14673:	14674:
U ODD8	14677:	14678:	14679:	14680:	14681:	14684:	14685:	14688:
U ODE0	14689:	14691:	14694:	14695:	14698:	14699:	14703:	14704:
U ODE8	14705:	14709:	14710:	14711:	14712:	14713:	14715:	14716:
U ODF0	14718:	14719:	14723:	14727:	14728:	14778:	14781:	14782:
U ODF8	14784:	14785:	14788:	14789:	14790:	14791:	14792:	14793:
U OE00	14797:	14798:	14799:	14800:	14801:	14806:	14807:	14808:
U OE08	14809:	14810:	14814:	15074:	15076:	15080:	15082:	15085:
U OE10	15088:	15089:	15093:	15095:	15096:	15098:	15099:	15102:
U OE18	15104:	15105:	15107:	15111:	15115:	15116:	15117:	15119:
U OE20	15120:	15122:	15124:	15125:	15128:	15129:	15132:	15133:
U OE28	15135:	15138:	15139:	15142:	15143:	15146:	15147:	15150:
U OE30	15153:	15155:	15156:	15157:	15158:	15159:	15226:	15227:
U OE38	15228:	15231:	15232:	15233:	15235:	15236:	15237:	15238:
U OE40	15297:	15298:	15299:	15301:	15302:	15303:	15307:	15308:
U OE48	15310:	15312:	15313:	15314:	15316:	15317:	15360:	15363:
U OE50	15364:	15366:	15367:	15370:	15371:	15372:	15373:	15374:
U OE58	15377:	15378:	15382:	15383:	15384:	15386:	15388:	15390:
U OE60	15391:	15394:	15395:	15398:	15399:	15402:	15403:	15404:
U OE68	15405:	15406:	15410:	15411:	15413:	15414:	15416:	15419:
U OE70	15420:	15423:	15424:	15426:	15427:	15428:	15432:	15433:
U OE78	15434:	15435:	15436:	15438:	15439:	15441:	15442:	15446:
U OF80	15450:	15451:	15501:	15504:	15505:	15507:	15508:	15511:
U OF88	15512:	15513:	15514:	15515:	15516:	15520:	15521:	15522:
U OF90	15523:	15524:	15528:	15529:	15530:	15531:	15532:	15535:
U OF98	15688:	15690:	15694:	15696:	15699:	15702:	15703:	15705:
U OFA0	15706:	15708:	15709:	15711:	15712:	15714:	15716:	15719:
U OFA8	15720:	15722:	15723:	15724:	15725:	15788:	15789:	15790:
U OFB0	15793:	15794:	15795:	15797:	15799:	15800:	15801:	15802:
U OFB8	15878:	15879:	15880:	15883:	15884:	15885:	15889:	15890:
U OFC0	15891:	15893:	15894:	15895:	15896:	15897:	15898:	15900:
U OFC8	15902:	15903:	15904:	15905:	15906:	15907:	15972:	15973:
U OFD0	15974:	15977:	15979:	15980:	15981:	15983:	15984:	15985:
U OFD8	15986:	16062:	16063:	16064:	16067:	16068:	16069:	16072:
U OFE0	16073:	16074:	16076:	16077:	16078:	16079:	16080:	16082:
U OFE8	16083:	16084:	16086:	16087:	16090:	16091:	16093:	16094:
U OFF0	16095:	16096:	16100:	16101:	16102:	16104:	16105:	16106:
U OFF8	16107:	16173:	16174:	16175:	16178:	16179:	16180:	16183:
U OF00	16184:	16185:	16186:	16262:	16263:	16264:	16267:	16268:
U OF08	16269:	16271:	16272:	16273:	16275:	16276:	16277:	16278:
U OF10	16279:	16280:	16283:	16286:	16287:	16290:	16291:	16292:
U OF18	16295:	16297:	16298:	16299:	16302:	16414:	16416:	16420:
U OF20	16422:	16425:	16428:	16429:	16432:	16433:	16435:	16436:
U OF28	16438:	16440:	16441:	16443:	16447:	16451:	16453:	16454:
U OF30	16455:	16456:	16458:	16462:	16463:	16464:	16465:	16466:

:Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U OF38	16467:	16468:	16469:	16470:	16552:	16553:	16554:	16557:
U OF40	16558:	16559:	16562:	16563:	16564:	16566:	16567:	16568:
U OF48	16569:	16574:	16575:	16576:	16578:	16579:	16581:	16584:
U OF50	16586:	16587:	16590:	16591:	16594:	16595:	16597:	16600:
U OF58	16601:	16604:	16605:	16608:	16609:	16612:	16615:	16617:
U OF60	16618:	16619:	16620:	16621:	16622:	16623:	16624:	16625:
U OF68	16707:	16708:	16709:	16712:	16713:	16714:	16717:	16718:
U OF70	16719:	16721:	16722:	16723:	16724:	16727:	16844:	16846:
U OF78	16850:	16852:	16855:	16858:	16859:	16862:	16863:	16865:
U OF80	16866:	16868:	16870:	16871:	16873:	16877:	16881:	16883:
U OF88	16884:	16885:	16886:	16888:	16892:	16893:	16895:	16896:
U OF90	16897:	16898:	16899:	16900:	16901:	16902:	16984:	16985:
U OF98	16986:	16989:	16990:	16991:	16994:	16995:	16996:	16998:
U OFA0	16999:	17000:	17001:	17003:	17006:	17007:	17009:	17010:
U OFA8	17011:	17012:	17013:	17014:	17015:	17097:	17098:	17099:
U OFB0	17102:	17103:	17104:	17107:	17108:	17109:	17111:	17112:
U OFB8	17113:	17114:	17116:	17219:	17221:	17225:	17227:	17230:
U OFC0	17233:	17234:	17237:	17238:	17240:	17241:	17243:	17247:
U OFC8	17248:	17250:	17251:	17254:	17255:	17256:	17257:	17258:
U OFD0	17350:	17351:	17352:	17355:	17356:	17357:	17360:	17361:
U OFD8	17362:	17364:	17365:	17366:	17367:	17369:	17372:	17373:
U OFE0	17375:	17376:	17379:	17380:	17381:	17382:	17475:	17476:
U OFE8	17477:	17480:	17481:	17482:	17485:	17486:	17487:	17489:
U OFF0	17490:	17491:	17492:	17494:	17739:	17741:	17745:	17747:
U OFF8	17750:	17753:	17754:	17757:	17758:	17760:	17761:	17763:
U 1000	17764:	17766:	17769:	17771:	17772:	17774:	17776:	17777:
U 1008	17779:	17780:	17781:	17782:	17783:	17846:	17847:	17848:
U 1010	17851:	17852:	17853:	17855:	17857:	17858:	17862:	17865:
U 1018	17866:	17868:	17870:	17871:	17872:	17874:	17875:	17877:
U 1020	17878:	17879:	17880:	17881:	17953:	17954:	17955:	17958:
U 1028	17959:	17960:	17963:	17964:	17965:	17967:	17969:	17970:
U 1030	17971:	17972:	17974:	17976:	17977:	17981:	17983:	17984:
U 1038	17986:	17988:	17989:	17990:	17993:	17994:	17996:	17997:
U 1040	17998:	17999:	18000:	18063:	18064:	18065:	18068:	18069:
U 1048	18070:	18072:	18073:	18074:	18076:	18077:	18149:	18150:
U 1050	18151:	18154:	18155:	18156:	18159:	18160:	18161:	18163:
U 1058	18164:	18165:	18167:	18168:	18169:	18170:	18172:	18173:
U 1060	18175:	18176:	18179:	18181:	18182:	18183:	18184:	18186:
U 1068	18188:	18189:	18190:	18191:	18193:	18194:	18197:	18198:
U 1070	18199:	18200:	18201:	18202:	18204:	18205:	18277:	18278:
U 1078	18279:	18282:	18283:	18284:	18287:	18288:	18289:	18292:
U 1080	18293:	18294:	18295:	18296:	18297:	18300:	18302:	18306:
U 1088	18307:	18310:	18311:	18313:	18315:	18316:	18318:	18321:
U 1090	18322:	18324:	18325:	18326:	18327:	18328:	18391:	18392:
U 1098	18393:	18396:	18397:	18398:	18401:	18402:	18403:	18405:
U 10A0	18406:	18478:	18479:	18480:	18483:	18484:	18485:	18488:
U 10A8	18489:	18490:	18493:	18494:	18495:	18496:	18497:	18499:
U 10B0	18501:	18502:	18503:	18506:	18507:	18508:	18509:	18510:
U 10B8	18573:	18574:	18575:	18578:	18579:	18580:	18582:	18583:
U 10C0	18585:	18589:	18590:	18662:	18663:	18664:	18667:	18668:
U 10C8	18669:	18672:	18673:	18674:	18677:	18678:	18679:	18680:
U 10D0	18681:	18682:	18683:	18686:	18851:	18853:	18857:	18859:
U 10D8	18862:	18865:	18866:	18869:	18870:	18872:	18873:	18875:

:Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 1238	20391:	20611:	20613:	20617:	20619:	20622:	20625:	20626:
U 1290	20629:	20630:	20632:	20633:	20636:	20637:	20638:	20639:
U 1298	20640:	20641:	20642:	20643:	20644:	20645:	20648:	20649:
U 12A0	20650:	20652:	20653:	20654:	20655:	20656:	20658:	20758:
U 12A8	20760:	20764:	20766:	20769:	20772:	20773:	20776:	20777:
U 12B0	20779:	20780:	20782:	20783:	20788:	20789:	20793:	20794:
U 12B8	20796:	20800:	20803:	20804:	20806:	20807:	20810:	20818:
U 12C0	20819:	20820:	20823:	20824:	20826:	20828:	20829:	20830:
U 12C8	20841:	20842:	20845:	20848:	20852:	20853:	20855:	20856:
U 12D0	20858:	20859:	20861:	20862:	20864:	20866:	20867:	20869:
U 12D8	20870:	20871:	20872:	20873:	20875:	20876:	20877:	20879:
U 12E0	20880:	20881:	20882:	20883:	20884:	20886:	20887:	20890:
U 12E8	20893:	20894:	20896:	20897:	20900:	20902:	20903:	20905:
U 12F0	20906:	20907:	20908:	20909:	20910:	20912:	20920:	20921:
U 12F8	20926:	20927:	20929:	20934:	21037:	21039:	21043:	21045:
U 1300	21048:	21051:	21052:	21055:	21056:	21058:	21059:	21061:
U 1308	21062:	21067:	21068:	21072:	21073:	21075:	21078:	21081:
U 1310	21082:	21085:	21086:	21090:	21098:	21099:	21100:	21103:
U 1318	21104:	21106:	21108:	21109:	21110:	21119:	21120:	21123:
U 1320	21126:	21130:	21131:	21133:	21134:	21136:	21137:	21139:
U 1328	21140:	21142:	21144:	21145:	21147:	21148:	21150:	21151:
U 1330	21152:	21155:	21156:	21157:	21159:	21160:	21161:	21162:
U 1338	21163:	21164:	21166:	21167:	21169:	21172:	21173:	21175:
U 1340	21176:	21179:	21181:	21182:	21184:	21185:	21186:	21187:
U 1348	21188:	21189:	21191:	21197:	21198:	21203:	21204:	21206:
U 1350	21210:	21360:	21362:	21366:	21368:	21371:	21374:	21375:
U 1358	21378:	21379:	21381:	21382:	21384:	21386:	21387:	21393:
U 1360	21394:	21396:	21397:	21400:	21401:	21403:	21404:	21406:
U 1368	21407:	21408:	21409:	21410:	21411:	21412:	21484:	21485:
U 1370	21486:	21489:	21490:	21491:	21494:	21495:	21496:	21498:
U 1378	21499:	21501:	21502:	21506:	21507:	21508:	21509:	21510:
U 1380	21514:	21515:	21516:	21517:	21518:	21522:	21523:	21524:
U 1388	21525:	21526:	21529:	21534:	21535:	21537:	21538:	21539:
U 1390	21541:	21542:	21543:	21544:	21624:	21625:	21626:	21629:
U 1398	21630:	21631:	21635:	21636:	21637:	21639:	21640:	21642:
U 13A0	21643:	21645:	21731:	21733:	21737:	21739:	21742:	21745:
U 13A8	21746:	21749:	21750:	21752:	21753:	21755:	21757:	21760:
U 13B0	21761:	21765:	21766:	21768:	21769:	21771:	21773:	21774:
U 13B8	21776:	21777:	21854:	21855:	21856:	21859:	21860:	21861:
U 13C0	21864:	21865:	21868:	21869:	21870:	21872:	21873:	21874:
U 13C8	21875:	21877:	21879:	21880:	21882:	22002:	22004:	22008:
U 13D0	22010:	22012:	22016:	22018:	22019:	22021:	22022:	22024:
U 13D8	22025:	22028:	22030:	22031:	22033:	22036:	22040:	22042:
U 13E0	22043:	22044:	22045:	22047:	22052:	22053:	22054:	22055:
U 13E8	22057:	22058:	22061:	22062:	22064:	22065:	22215:	22217:
U 13F0	22218:	22220:	22221:	22223:	22224:	22225:	22226:	22230:
U 13F8	22231:	22232:	22234:	22236:	22237:	22238:	22240:	22241:
U 1400	22242:	22244:	22245:	22248:	22249:	22250:	22252:	22253:
U 1408	22254:	22255:	22258:	22365:	22367:	22371:	22373:	22375:
U 1410	22378:	22379:	22381:	22382:	22384:	22385:	22388:	22390:
U 1418	22391:	22393:	22396:	22400:	22402:	22403:	22404:	22405:
U 1420	22407:	22411:	22412:	22413:	22415:	22416:	22418:	22421:
U 1428	22422:	22423:	22425:	22426:	22428:	22431:	22432:	22433:

;Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 1430	22435:	22436:	22439:	22442:	22443:	22444:	22446:	22447:
U 1438	22449:	22452:	22453:	22454:	22456:	22457:	22460:	22558:
U 1440	22560:	22564:	22566:	22569:	22572:	22573:	22576:	22577:
U 1448	22579:	22583:	22584:	22586:	22587:	22588:	22589:	22590:
U 1450	22591:	22648:	22649:	22650:	22653:	22654:	22655:	22659:
U 1458	22660:	22661:	22663:	22664:	22665:	22666:	22673:	22674:
U 1460	22676:	22677:	22678:	22679:	22680:	22682:	22683:	22684:
U 1468	22685:	22690:	22691:	22692:	22693:	22694:	22695:	22808:
U 1470	22809:	22810:	22813:	22814:	22815:	22818:	22819:	22820:
U 1478	22822:	22823:	22824:	22825:	22829:	22960:	22962:	22966:
U 1480	22968:	22971:	22974:	22975:	22979:	22980:	22982:	22985:
U 1488	22985:	22988:	22989:	22990:	22991:	22992:	22993:	23051:
U 1490	23052:	23053:	23056:	23057:	23058:	23061:	23062:	23063:
U 1498	23064:	23065:	23066:	23067:	23072:	23073:	23076:	23077:
U 14A0	23078:	23079:	23084:	23085:	23086:	23087:	23088:	23089:
U 14A8	23185:	23186:	23187:	23190:	23191:	23192:	23195:	23196:
U 14B0	23197:	23198:	23199:	23200:	23201:	23203:	23205:	23206:
U 14B8	23208:	23318:	23320:	23324:	23326:	23329:	23332:	23333:
U 14C0	23337:	23338:	23341:	23344:	23345:	23346:	23347:	23348:
U 14C8	23350:	23351:	23353:	23354:	23355:	23356:	23357:	23361:
U 14D0	23362:	23363:	23364:	23365:	23366:	23468:	23469:	23470:
U 14D8	23473:	23474:	23475:	23478:	23479:	23480:	23482:	23483:
U 14E0	23484:	23485:	23487:	23488:	23493:	23495:	23496:	23498:
U 14E8	23499:	23502:	23503:	23504:	23505:	23508:	23509:	23510:
U 14F0	23511:	23512:	23513:	23594:	23595:	23596:	23599:	23600:
U 14F8	23601:	23605:	23606:	23607:	23609:	23610:	23611:	23612:
U 1500	23614:	23615:	23616:	23620:	23770:	23772:	23776:	23778:
U 1508	23781:	23784:	23785:	23789:	23790:	23793:	23798:	23799:
U 1510	23801:	23802:	23803:	23804:	23805:	23806:	23807:	23865:
U 1518	23866:	23867:	23870:	23871:	23872:	23875:	23876:	23877:
U 1520	23879:	23880:	23881:	23882:	23884:	23885:	23887:	23889:
U 1528	23890:	23891:	23892:	23893:	23895:	23896:	23897:	23898:
U 1530	23903:	23904:	23905:	23906:	23907:	24020:	24021:	24022:
U 1538	24025:	24026:	24027:	24031:	24032:	24033:	24035:	24036:
U 1540	24037:	24038:	24044:	24046:	24048:	24049:	24051:	24052:
U 1548	24055:	24056:	24058:	24059:	24062:	24063:	24064:	24065:
U 1550	24070:	24071:	24072:	24073:	24074:	24155:	24156:	24157:
U 1558	24160:	24161:	24162:	24165:	24166:	24167:	24169:	24170:
U 1560	24171:	24172:	24175:	24176:	24179:	24180:	24181:	24182:
U 1568	24183:	24184:	24185:	24190:	24191:	24192:	24193:	24194:
U 1570	24288:	24294:	24295:	24296:	24297:	24300:	24301:	24302:
U 1578	24304:	24305:	24306:	24307:	24309:	24438:	24440:	24444:
U 1580	24446:	24449:	24452:	24453:	24457:	24458:	24461:	24463:
U 1588	24464:	24466:	24469:	24473:	24475:	24476:	24477:	24479:
U 1590	24481:	24486:	24487:	24488:	24489:	24490:	24491:	24494:
U 1598	24495:	24497:	24498:	24499:	24500:	24501:	24502:	24560:
U 15A0	24561:	24562:	24565:	24566:	24567:	24569:	24570:	24571:
U 15A8	24572:	24576:	24577:	24578:	24579:	24580:	24581:	24582:
U 15B0	24676:	24677:	24678:	24681:	24682:	24683:	24686:	24687:
U 15B8	24688:	24690:	24691:	24692:	24693:	24695:	24696:	24697:
U 15C0	24702:	24704:	24705:	24707:	24708:	24709:	24710:	24712:
U 15C8	24714:	24716:	24717:	24718:	24719:	24722:	24723:	24724:
U 15D0	24725:	24726:	24727:	24728:	24822:	24823:	24824:	24827:

:Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 15D8	24828:	24829:	24833:	24834:	24835:	24837:	24838:	24839:
U 15E0	24840:	24842:	24843:	24845:	24961:	24963:	24967:	24969:
U 15E8	24972:	24975:	24976:	24979:	24980:	24983:	24984:	24991:
U 15F0	24993:	24994:	24997:	25003:	25005:	25006:	25007:	25009:
U 15F8	25011:	25014:	25016:	25017:	25018:	25019:	25020:	25022:
U 1600	25023:	25024:	25025:	25029:	25030:	25031:	25032:	25033:
U 1608	25034:	25136:	25137:	25138:	25141:	25142:	25143:	25146:
U 1610	25148:	25153:	25154:	25155:	25156:	25157:	25158:	25271:
U 1618	25272:	25273:	25276:	25277:	25278:	25281:	25285:	25286:
U 1620	25287:	25288:	25292:	25293:	25294:	25295:	25296:	25297:
U 1628	25298:	25400:	25401:	25402:	25405:	25406:	25407:	25410:
U 1630	25411:	25414:	25415:	25416:	25418:	25419:	25420:	25421:
U 1638	25422:	25424:	25549:	25551:	25555:	25557:	25560:	25563:
U 1640	25564:	25567:	25568:	25570:	25571:	25573:	25575:	25576:
U 1648	25578:	25581:	25585:	25587:	25588:	25589:	25590:	25592:
U 1650	25595:	25596:	25599:	25600:	25601:	25602:	25703:	25704:
U 1658	25705:	25708:	25709:	25710:	25713:	25715:	25717:	25718:
U 1660	25719:	25720:	25722:	25726:	25727:	25730:	25731:	25732:
U 1668	25733:	25734:	25927:	25928:	25929:	25932:	25933:	25934:
U 1670	25937:	25938:	25939:	25941:	25942:	25943:	25944:	25945:
U 1678	25950:	25951:	25954:	25955:	25956:	25957:	26058:	26059:
U 1680	26060:	26063:	26064:	26065:	26067:	26173:	26175:	26179:
U 1688	26181:	26184:	26187:	26188:	26191:	26192:	26194:	26195:
U 1690	26197:	26199:	26200:	26202:	26205:	26209:	26211:	26212:
U 1698	26213:	26214:	26216:	26219:	26221:	26223:	26224:	26225:
U 16A0	26226:	26228:	26232:	26233:	26236:	26237:	26238:	26239:
U 16A8	26240:	26432:	26433:	26434:	26437:	26438:	26439:	26443:
U 16B0	26444:	26448:	26449:	26452:	26453:	26454:	26455:	26556:
U 16B8	26557:	26558:	26561:	26562:	26563:	26566:	26567:	26568:
U 16C0	26570:	26571:	26572:	26573:	26575:	26716:	26718:	26722:
U 16C8	26724:	26727:	26730:	26731:	26734:	26735:	26737:	26738:
U 16D0	26740:	26742:	26743:	26745:	26748:	26752:	26754:	26755:
U 16D8	26756:	26757:	26759:	26763:	26764:	26766:	26767:	26768:
U 16E0	26769:	26870:	26871:	26872:	26875:	26876:	26877:	26879:
U 16E8	26880:	26882:	26883:	26884:	26885:	26887:	26891:	26892:
U 16F0	26894:	26895:	26896:	26897:	26898:	27091:	27092:	27093:
U 16F8	27096:	27097:	27098:	27101:	27102:	27103:	27105:	27106:
U 1700	27107:	27108:	27110:	27115:	27116:	27118:	27119:	27120:
U 1708	27121:	27222:	27223:	27224:	27227:	27228:	27229:	27232:
U 1710	27233:	27235:	27236:	27237:	27238:	27240:	27244:	27245:
U 1718	27247:	27248:	27249:	27250:	27251:	27444:	27445:	27446:
U 1720	27449:	27450:	27451:	27454:	27455:	27456:	27458:	27459:
U 1728	27460:	27461:	27462:	27466:	27467:	27469:	27470:	27471:
U 1730	27472:	27573:	27574:	27575:	27578:	27579:	27580:	27583:
U 1738	27707:	27709:	27713:	27715:	27718:	27721:	27722:	27725:
U 1740	27726:	27728:	27729:	27731:	27733:	27734:	27736:	27739:
U 1748	27743:	27745:	27746:	27747:	27748:	27750:	27753:	27754:
U 1750	27755:	27759:	27760:	27763:	27764:	27765:	27766:	27867:
U 1758	27868:	27869:	27872:	27873:	27874:	27876:	27877:	27879:
U 1760	27880:	27881:	27882:	27884:	27888:	27889:	27891:	27892:
U 1768	27893:	27894:	27895:	28088:	28089:	28090:	28093:	28094:
U 1770	28095:	28098:	28099:	28100:	28102:	28103:	28104:	28105:
U 1778	28106:	28110:	28111:	28115:	28119:	28120:	28121:	28123:

;Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 1780	28126:	28127:	28129:	28130:	28131:	28132:	28233:	28234:
U 1788	28235:	28238:	28239:	28240:	28243:	28364:	28366:	28370:
U 1790	28372:	28375:	28378:	28379:	28382:	28383:	28385:	28386:
U 1798	28388:	28389:	28391:	28393:	28394:	28396:	28399:	28403:
U 17A0	28405:	28406:	28407:	28408:	28410:	28416:	28417:	28418:
U 17A8	28419:	28420:	28424:	28425:	28428:	28429:	28430:	28431:
U 17B0	28532:	28533:	28534:	28537:	28538:	28539:	28543:	28544:
U 17B8	28545:	28546:	28547:	28549:	28550:	28554:	28555:	28558:
U 17C0	28559:	28560:	28561:	28562:	28563:	28564:	28777:	28778:
U 17C8	28779:	28782:	28783:	28784:	28787:	28788:	28789:	28791:
U 17D0	28792:	28794:	28796:	28797:	28799:	28801:	28802:	28805:
U 17D8	28806:	28809:	28810:	28811:	28812:	28813:	28817:	28818:
U 17E0	28819:	28820:	28821:	28824:	28825:	28828:	28829:	28830:
U 17E8	28831:	28932:	28933:	28934:	28937:	28938:	28939:	28941:
U 17F0	29096:	29098:	29102:	29104:	29107:	29110:	29111:	29113:
U 17F8	29114:	29117:	29118:	29120:	29121:	29124:	29129:	29130:
U 1800	29131:	29133:	29134:	29135:	29136:	29137:	29141:	29142:
U 1808	29144:	29145:	29146:	29147:	29150:	29151:	29154:	29156:
U 1810	29157:	29158:	29159:	29160:	29161:	29162:	29242:	29243:
U 1818	29244:	29247:	29248:	29249:	29252:	29253:	29254:	29256:
U 1820	29257:	29258:	29259:	29260:	29261:	29263:	29265:	29266:
U 1828	29268:	29269:	29271:	29273:	29274:	29276:	29277:	29278:
U 1830	29282:	29283:	29285:	29286:	29287:	29290:	29291:	29292:
U 1838	29295:	29296:	29299:	29300:	29303:	29304:	29305:	29307:
U 1840	29308:	29309:	29310:	29311:	29312:	29314:	29318:	29319:
U 1848	29324:	29329:	29330:	29333:	29334:	29335:	29336:	29337:
U 1850	29417:	29418:	29419:	29422:	29423:	29424:	29426:	29428:
U 1858	29429:	29431:	29433:	29436:	29437:	29438:	29440:	29441:
U 1860	29442:	29443:	29446:	29536:	29538:	29542:	29544:	29547:
U 1868	29550:	29551:	29554:	29555:	29557:	29558:	29560:	29561:
U 1870	29564:	29569:	29573:	29574:	29577:	29578:	29579:	29580:
U 1878	29581:	29661:	29662:	29663:	29666:	29667:	29668:	29671:
U 1880	29673:	29674:	29678:	29680:	29681:	29683:	29684:	29687:
U 1888	29688:	29689:	29692:	29693:	29694:	29698:	29700:	29701:
U 1890	29702:	29703:	29706:	29707:	29709:	29711:	29712:	29714:
U 1898	29715:	29717:	29718:	29720:	29722:	29726:	29727:	29729:
U 18A0	29730:	29732:	29733:	29735:	29737:	29740:	29741:	29743:
U 18A8	29745:	29746:	29751:	29752:	29756:	29757:	29760:	29761:
U 18B0	29762:	29763:	29764:	29844:	29845:	29846:	29849:	29850:
U 18B8	29851:	29853:	29855:	29856:	29859:	29860:	29861:	29864:
U 18C0	29865:	29866:	29868:	29869:	29870:	29871:	29872:	29874:
U 18C8	29979:	29981:	29985:	29987:	29990:	29993:	29994:	29997:
U 18D0	29998:	30000:	30001:	30003:	30006:	30007:	30011:	30012:
U 18D8	30013:	30014:	30115:	30116:	30117:	30120:	30121:	30122:
U 18E0	30127:	30128:	30131:	30132:	30133:	30134:	30135:	30136:
U 18E8	30238:	30239:	30240:	30243:	30244:	30245:	30246:	30247:
U 18F0	30250:	30251:	30252:	30254:	30255:	30256:	30257:	30260:
U 18F8	30262:	30263:	30266:	30268:	30271:	30272:	30273:	30275:
U 1900	30276:	30277:	30278:	30280:	30285:	30286:	30289:	30290:
U 1908	30291:	30292:	30293:	30294:	30396:	30397:	30398:	30401:
U 1910	30402:	30403:	30404:	30405:	30408:	30409:	30410:	30414:
U 1918	30415:	30416:	30418:	30419:	30420:	30421:	30423:	30523:
U 1920	30525:	30529:	30531:	30534:	30537:	30538:	30541:	30542:

ZZ-ENKCF-1.4 ;
: ENKCF.MCR
:

Location / Line Num15-MAR-83
MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module
Location / Line Number Index

C 7
Fiche 4 Frame C7

Sequence 698
Rev 01.40 Page 692

:Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 1928	30544:	30545:	30547:	30551:	30552:	30556:	30557:	30560:
U 1930	30561:	30562:	30563:	30564:	30565:	30667:	30668:	30669:
U 1938	30672:	30673:	30674:	30677:	30678:	30679:	30681:	30682:
U 1940	30683:	30684:	30686:	30689:	30690:	30691:	30693:	30697:
U 1948	30698:	30701:	30702:	30703:	30704:	30796:	30797:	30798:
U 1950	30801:	30802:	30803:	30806:	30807:	30808:	30810:	30811:
U 1958	30812:	30813:	30815:	30908:	30910:	30914:	30916:	30919:
U 1960	30922:	30923:	30926:	30927:	30929:	30930:	30932:	30935:
U 1968	30936:	30940:	30941:	30942:	30943:	31044:	31045:	31046:
U 1970	31049:	31050:	31051:	31056:	31057:	31058:	31059:	31124:
U 1978	31125:	31126:	31129:	31130:	31131:	31132:	31133:	31136:
U 1980	31137:	31138:	31140:	31141:	31142:	31143:	31145:	31147:
U 1988	31148:	31151:	31153:	31156:	31157:	31158:	31160:	31161:
U 1990	31162:	31163:	31165:	31170:	31171:	31172:	31173:	31238:
U 1998	31239:	31240:	31243:	31244:	31245:	31246:	31247:	31250:
U 19A0	31251:	31252:	31255:	31256:	31257:	31259:	31260:	31261:
U 19A8	31262:	31264:	31364:	31366:	31370:	31372:	31375:	31378:
U 19B0	31379:	31382:	31383:	31385:	31386:	31389:	31392:	31393:
U 19B8	31396:	31397:	31398:	31399:	31501:	31502:	31503:	31508:
U 19C0	31509:	31510:	31514:	31515:	31516:	31518:	31519:	31520:
U 19C8	31521:	31523:	31526:	31527:	31530:	31531:	31532:	31533:
U 19D0	31534:	31636:	31637:	31638:	31641:	31642:	31643:	31646:
U 19D8	31647:	31648:	31650:	31651:	31652:	31653:	31655:	31858:
U 19E0	31860:	31864:	31866:	31869:	31873:	31874:	31876:	31878:
U 19E8	31881:	31882:	31884:	31887:	31888:	31891:	31892:	31893:
U 19F0	31894:	31895:	31896:	31897:	31898:	31957:	31958:	31959:
U 19F8	31962:	31963:	31964:	31965:	31966:	31967:	31968:	31969:
U 1A00	31970:	31971:	31972:	31973:	31974:	31975:	31976:	31977:
U 1A08	31979:	31980:	31981:	31982:	31983:	31984:	31985:	31986:
U 1A10	31987:	31988:	31990:	31991:	31993:	31994:	31996:	31998:
U 1A18	31999:	32000:	32001:	32003:	32006:	32007:	32008:	32009:
U 1A20	32012:	32015:	32017:	32021:	32022:	32024:	32025:	32026:
U 1A28	32027:	32028:	32031:	32032:	32033:	32034:	32037:	32038:
U 1A30	32039:	32040:	32041:	32042:	32043:	32044:	32045:	32046:
U 1A38	32047:	32048:	32050:	32051:	32052:	32053:	32055:	32062:
U 1A40	32063:	32064:	32065:	32066:	32067:	32132:	32133:	32134:
U 1A48	32137:	32138:	32139:	32140:	32141:	32142:	32143:	32144:
U 1A50	32145:	32146:	32147:	32148:	32149:	32150:	32153:	32154:
U 1A58	32155:	32156:	32158:	32162:	32164:	32168:	32169:	32174:
U 1A60	32175:	32176:	32177:	32178:	32179:	32238:	32239:	32240:
U 1A68	32243:	32244:	32245:	32246:	32247:	32248:	32249:	32250:
U 1A70	32251:	32252:	32253:	32254:	32255:	32258:	32259:	32260:
U 1A78	32261:	32263:	32267:	32268:	32269:	32270:	32271:	32272:
U 1A80	32273:	32274:	32333:	32334:	32335:	32338:	32339:	32340:
U 1A88	32341:	32342:	32343:	32344:	32347:	32348:	32349:	32350:
U 1A90	32351:	32352:	32353:	32354:	32355:	32356:	32357:	32358:
U 1A98	32361:	32362:	32363:	32364:	32366:	32371:	32373:	32377:
U 1AA0	32378:	32379:	32384:	32386:	32390:	32391:	32393:	32394:
U 1AA8	32395:	32396:	32487:	32489:	32493:	32495:	32498:	32501:
U 1AB0	32502:	32505:	32506:	32508:	32510:	32513:	32514:	32516:
U 1AB8	32518:	32521:	32522:	32524:	32525:	32528:	32529:	32534:
U 1AC0	32535:	32541:	32542:	32543:	32544:	32547:	32548:	32549:
U 1AC8	32550:	32551:	32552:	32553:	32554:	32555:	32635:	32636:

;Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 1AD0	32637:	32640:	32641:	32642:	32643:	32644:	32645:	32646:
U 1AD8	32648:	32650:	32651:	32652:	32653:	32654:	32656:	32659:
U 1AE0	32660:	32661:	32662:	32663:	32666:	32667:	32670:	32671:
U 1AE8	32673:	32675:	32676:	32679:	32680:	32684:	32685:	32687:
U 1AF0	32692:	32693:	32696:	32697:	32698:	32699:	32700:	32701:
U 1AF8	32702:	32703:	32704:	32784:	32785:	32786:	32789:	32790:
U 1B00	32791:	32792:	32793:	32794:	32795:	32797:	32799:	32800:
U 1B08	32803:	32804:	32805:	32807:	32810:	32811:	32814:	32815:
U 1B10	32816:	32817:	32818:	32819:	32822:	32823:	32824:	32825:
U 1B18	32826:	32829:	32830:	32833:	32834:	32836:	32971:	32973:
U 1B20	32977:	32979:	32982:	32985:	32986:	32990:	32992:	32993:
U 1B28	32995:	32997:	32998:	33000:	33001:	33003:	33004:	33006:
U 1B30	33007:	33009:	33010:	33014:	33015:	33017:	33019:	33021:
U 1B38	33022:	33023:	33025:	33026:	33027:	33028:	33029:	33030:
U 1B40	33031:	33033:	33034:	33035:	33039:	33040:	33041:	33042:
U 1B48	33045:	33047:	33048:	33049:	33050:	33053:	33055:	33056:
U 1B50	33057:	33058:	33061:	33063:	33064:	33065:	33066:	33069:
U 1B58	33071:	33072:	33073:	33074:	33076:	33180:	33182:	33186:
U 1B60	33188:	33191:	33194:	33195:	33199:	33200:	33205:	33209:
U 1B68	33211:	33214:	33215:	33218:	33219:	33220:	33223:	33224:
U 1B70	33253:	33254:	33255:	33257:	33258:	33259:	33260:	33262:
U 1B78	33263:	33264:	33265:	33266:	33267:	33268:	33270:	33271:
U 1B80	33272:	33276:	33278:	33280:	33284:	33286:	33287:	33294:
U 1B88	33295:	33296:	33298:	33299:	33300:	33301:	33303:	33305:
U 1B90	33308:	33310:	33314:	33315:	33316:	33319:	33320:	33323:
U 1B98	33324:	33325:	33327:	33328:	33329:	33330:	33332:	33420:
U 1BA0	33422:	33426:	33428:	33431:	33434:	33435:	33438:	33439:
U 1BA8	33441:	33442:	33446:	33448:	33450:	33452:	33453:	33456:
U 1BB0	33459:	33460:	33465:	33467:	33469:	33472:	33473:	33474:
U 1BB8	33475:	33476:	33555:	33556:	33557:	33560:	33561:	33562:
U 1BC0	33563:	33565:	33568:	33569:	33572:	33573:	33574:	33576:
U 1BC8	33577:	33578:	33579:	33581:	33584:	33588:	33590:	33593:
U 1BD0	33594:	33597:	33600:	33601:	33605:	33607:	33609:	33612:
U 1BD8	33613:	33614:	33615:	33616:	33695:	33696:	33697:	33700:
U 1BE0	33701:	33702:	33703:	33705:	33708:	33709:	33712:	33713:
U 1BE8	33714:	33716:	33717:	33718:	33719:	33722:	33729:	33730:
U 1BF0	33732:	33734:						

ZZ-ENKCF-1.4 :
: ENKCF.MCR
:

E 7
C-code Microword Su15-MAR-83
MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module
C-code Microword Summary

Fiche 4 Frame E7

Sequence 700

Rev 01.40

Page 694

Words not
in bounds
ENKCF 512

Used 512
Remaining

Total microwords used in memory C: 512
Total microwords remaining in memory C: 0
Highest address used in memory C: 180 (hex)

Words not
in bounds
ENKCF 6206

Used 6206
Remaining

Total microwords used in memory U: 6206
Total microwords remaining in memory U: 0
Highest address used in memory U: 1BF1 (hex)

ZZ-ENKCF-1.4 ;
: ENKCF.MCR ;
: Summary
MICRO2 1M(01) 15-MAR-83 11:46:22 ENKCF Micro-diagnostic for IDC module Rev 01.40
Summary

: The file used in this assembly is:
ENKCF.MIC

Pass 1 warnings:	0	Pass 2 warnings:	0
Pass 1 errors:	0	Pass 2 errors:	0